Online Programming Tutors or Paper Study Guides?

Wendy Fisher, Cyndi Rader, Tracy Camp
Department of Electrical Engineering and Computer Science
Colorado School of Mines
Golden, CO USA
wbelcher@mines.edu, crader@mines.edu, tcamp@mines.edu

***Abstract*—An undergraduate data structures course is challenging to teach due to the vast number of complex topics that need to be covered. The use of instructional tools, such as an online programming tutor, can reinforce topics students typically find difficult. Prior research has shown using programming tutors can have a positive impact on student learning in an introductory Computer Science course. We hypothesized that we would see similar results using an online tutor in a second year programming course. Our study used an existing programming tutor, Problets, to supplement course instruction on two identified topics (functions and pointers) over one semester in two sections of a data structures course. In the first part of the study, we compared the optional use of an online tutor to no supplemental instruction. In the second part of the study, we compared the use of the online tutor to paper study guides. We conducted a two-tailed unpaired t-test on the students' midterm examination scores. The results from the first module showed that simply providing an optional tool had limited value. Surprisingly, the results from the second module showed no significant difference based on the type of practice. We discuss possible explanations for these results.**

Keywords - online programming tutor, data structures, evaluation, computer science education, incentives, study guide

I. Introduction

An undergraduate data structures course is challenging to teach due to the vast number of complex topics that need to be covered during one semester. The use of instructional tools, such as an online programming tutor, can reinforce concepts students typically struggle with. The advantages of an online system over the traditional book and paper assignments include instant feedback and grading, and the ability to easily attempt multiple problems of the same type. Prior research shows the use of programming tutors has a positive impact on student learning in an introductory Computer Science course [1]. Our study used an existing online programming tutor, Problets [2], to supplement course instruction on two identified topics (functions and pointers) in a data structures course. We hypothesized that we would see similar results as previously found, i.e., positive impact on student learning in a data structures course due to a programming tutor.

Problets are a set of online programming tutors developed, and used in many research studies over the last 14 years, by Amruth N. Kumar at the Ramapo College of New Jersey [2]. These problem solving software assistants are intended as supplemental tools to complement traditional class instruction and projects in an introductory programming course. They are adaptive learning systems that provide focused practice on several key concepts, such as expression evaluation, selection statements, loops, functions, arrays, and pointers. The Problets system generates problems adaptively from a large repository of templates, automatically grades student answers, provides concise feedback, has a textual and graphic interface, and logs all the data for future analysis. Research conducted on the effectiveness of Problets has focused on introductory programming courses [3], mainly on overall and gender differences [4]–[6] in attitudes, and the performance of the tutor [7], [8]. Research results have been primarily measured using pre-test, practice, post-test results. Our study aims to determine whether there is a measurable effect on the use of online programming tutors in a second year programming course. Does an online programming tutor improve students' understanding and performance on specific examination topics? We describe the study, our methods, the interface, pre-test, the Problets tutor selected for each of our two experiments, and our post attitudinal survey. We also provide the results and discussion from our evaluation.

II. Background and Related Work

Research studies have identified a number of instructional methods that can effectively enhance student learning when used to supplement course instruction. In addition to the online software-programming tutors discussed in our study, the use of visual tools, programming competitions, and games have been found to be effective. This section highlights the use of programming tutors in introductory Computer Science courses, as well as differences in student attitudes. We also review the effectiveness of a few other supplemental instructional methods that have been used for teaching complex concepts in a data structures course.

Many studies have been published on the performance of the individual Problets' modules. For example, Kumar evaluated the dynamic memory module and found there was improvement between the pre-test and post-test results [9]. Kumar and Kaczmarczyk presented findings of a study on gender and ethnicity differences in student learning using an online programming tutor [10]. Overall, the study concluded students display the same need and benefit from the programming tutors examined, regardless of gender or ethnicity.

Kumar also studied differences in affective learning (motivation, attitudes, and emotions) of students when introduced to online software tutors during an introductory Computer Science course [11]. The author analyzed the results of two

different tutors and found that although male students and traditionally represented students were better prepared than the others, the Likert-scale surveys found no significant difference in affective learning on either of the programming tutors between demographic groups.

Lawrence investigated the student motivation in an introductory data structures class using a competitive gaming project [12]. The author described a friendly student competition targeted to motivate students and increase the success rate of learning core skills for C++ programming (e.g., lists, stacks, queues, and trees). The implementation of a strategy game, such as checkers or chess, in a real-time competitive tournament environment was assigned as the final project. In general, the introduction of a competitive gaming project motivated students and made the programming assignment more interesting.

García-Mateos and Fernández-Alemán examined student dropout rate and motivation through the introduction of online judging and competitions for programming projects in a data structures course [13]. The authors replaced the final examination with a gaming project that was integrated with an open-source automatic judging software called Mooshak [14]. An attitudinal survey supported their hypothesis that a continuous evaluation environment increased student's motivation over the traditional format; in fact 91% of students stated they would follow the new strategy again.

Patel evaluated different tools that have been used for learning data structures and provided a comparison of their effectiveness [15]. The study aimed to identify the types of tools and techniques that are most effective for solving the common problems of low motivation and understanding abstract concepts. The author compared the three most popular tools currently being used to enhance student learning in a data structures course: Multimedia-interactive systems, Learning through Graphics, and Game-based learning. The game-based learning style (see [12] for details) was rated the most effective of all three methods to supplement course instruction in teaching data structures by providing fun, interesting, and challenging competitive projects.

III. Methods

Our research question is: "does the use of an online programming tutor have a measurable effect on student performance on topic understanding in a data structures course?" We collected information from students enrolled in two sections of a C++ data structures course during the Spring 2015 semester. To retain consistency, students received the same instruction from the course professor and took the same midterm examinations.

We focused on two specific C++ data structure concepts: functions and pointers. The two topics selected aligned with material included in the first and second midterms (functions and pointers, respectively). Note that students do receive instruction on functions within a required prerequisite introductory programming class, but there are many subtleties that students do not master during that initial course. The tutoring session, including pre-test and practice, required approximately 45 minutes or less for each student to complete. Only those students who submitted a consent form (N=55) have their results confidentially included in our study; results from one minor student are omitted.

A. The Interface

We created an online user interface for students to use with our contact information, instructions, and a method to opt out. The interface also included links to the consent form, pre-test, help, and access to the specific topics selected from the online programming tutor (i.e., either functions or pointers).

B. Module 1: Functions

The functions module was deployed approximately midway between the lecture on functions, parameters, and return types and the first midterm examination. The students received the assignment three weeks after class discussion, were given one week to complete, and then had one week before the first midterm. The two sections (A and B) were split into experimental and control groups. Students in Section A did the pre-test and were then asked to do the programming tutor; students in Section B only did the pre-test.

1) Incentive: Students were offered five points of extra credit as an incentive for their participation. Only the students who completed the functions online tutor received the extra points.

2) Pre-test: Following class instruction on the topic of functions, all students completed a 5-question pre-test with no feedback. The questions were modeled to cover similar learning objectives as in the functions online tutor; two example questions are listed in Table I.

TABLE I
EXAMPLE FUNCTIONS PRE-TEST QUESTIONS

Where does a function need to be defined / prototyped?

a. Before it is called.

b. After it is called.

c. Does not matter, as long as it is in the same file.

What is the output from the following code sample?

```
void swap (int &a, int b)
{
   int temp = a;
   a = b;
   b = temp;
}

int main()
{
   int myA = 4, myB = 8;
   swap(myA, myB);
   cout << "Swapped values: " << myA;
   cout << " and "  << myB << endl;
}
```

The remaining questions on the functions pre-test covered topic scenarios such as:

- Return not the last statement in the function.
- Data type of return statement doesn't match function.
- Data type of parameter doesn't match definition.
- Using value returned by function with void return type.

The results of the pre-test were used to identify those already proficient in the topic, and to ensure there were no significant initial differences in content knowledge between experimental and control groups. Students receiving a 100% on the pre-test, if any, would be eliminated from the study.

3) The Tutor: In the first module of our study, we used the Problets *Debugging Functions* tutor. The learning objectives include topics to help students understand the purpose and use of functions in C++. Through a series of debugging exercises, students learn how to declare and define functions, calling rules, the proper use of return types and parameters, and scoping rules for local function variables.

C. Module 2: Pointers

The pointers module was deployed approximately mid-way between the lecture on pointers and dynamic memory and the second midterm examination. Based on the low participation on the first module (functions), we created a required assignment that also allowed us to explore whether the online tutoring system provided more benefit than a traditional paper study guide. The addition of the study guide allowed for a comparison of students who spent equivalent amounts of time on two different reinforcement activities. The study guide had nine questions that replicated the primary questions in the pointers online tutor and was developed to take the same estimated time to complete. The main difference was the study guide did not have the visual interface, feedback system, or additional practice problems that the online system had.

The students received the assignment four weeks after class discussion, were given one week to complete, and then had two weeks before the second midterm. The pointers module was required for all students in both sections (though only results from students with consent are included). To avoid self-selection, the students were randomly split such that 50% were assigned the online version and 50% were assigned the paper study guide. The study guide was not graded or returned; however, the solutions were posted before the examination on the course website.

1) Incentive: Since the pointers activity was required for all students in both sections, no additional incentive was offered. The assignment was worth a total of 20 points, which is about half of a lab grade. There was a total of 180 possible lab points to earn for the semester.

2) Pre-test: Following class instruction on the topic of pointers, all students completed a 5-question pre-test with no feedback. The questions were structured the same as the functions pre-test and modeled to cover similar learning objectives as in the online tutor; an example question is listed in Table II.

TABLE II
EXAMPLE POINTERS PRE-TEST QUESTION

Which one of the following would cause a syntax error?

a. int *intPtr;

b. string s, *strPtr = 0;

c. int myInt; double* dp = &myInt;

d. int *pi = 0;

e. None of the above, they are all legal statements.

The remaining questions on the pointers pre-test covered topic scenarios such as:

- Incorrect dereferencing or indirection.
- Pointers, arrays, and memory locations.
- Memory leaks and dangling pointers.

As with the first module (functions), the results of the pre-test were used to identify those already proficient in the topic, and to ensure there were no significant initial differences in content knowledge between experimental and control groups.

3) The Tutor: In the second module of our study, we used the Problets *C++ Pointers* tutor. The learning objectives include topics to help students understand the purpose and use of pointers and dynamic memory in C++. Given a series of coding examples, students learn how to identify memory management issues, such as memory leaks and dangling pointers. Students also gain experience with accessing a pointer out of its scope or referencing pointers to uninitialized variables. After taking the pre-test in class, students were given one week to complete the tutor (or paper study guide) on their own time.

IV. RESULTS AND ANALYSIS

After students completed the two selected modules, their performance was evaluated. Only the questions specific to the topics being analyzed were used (not the overall test score). We conducted a two-tailed unpaired t-test on the students' examination scores to determine the effectiveness of using an online programming tutor as a supplement for class instruction.

A. Module 1: Functions

For this module, the experimental group consisted of students assigned the Problets tutor on functions and the control group consisted of students not assigned anything additional.

1) Pre-test Results: We examined the scores from the students who completed the pre-test (N=54). The averages were 56.36% for the experimental (tutor) group and 50.53% for the control (no tutor) group. No student received a 100%, which indicates that students were not completely proficient in the learning objectives covered in the programming tutor.

2) Midterm Scores Analysis: We developed three midterm examination questions that focused on the specific topics covered in the *Debugging Functions* tutor; each question was worth 5 points and they contributed to 15% of the overall test score. In the experimental group, only 11 students completed the online tutor, so our results are only based on the 11 students along with the full control group.

The average score was 87.88% for those who did use the tutor (experimental) and 80.70% for those who did not use a tutor (control). These results indicate most students correctly answered 2 of the 3 questions correctly. The distribution for the average scores of the three functions questions on the first midterm is shown in Figure 1.

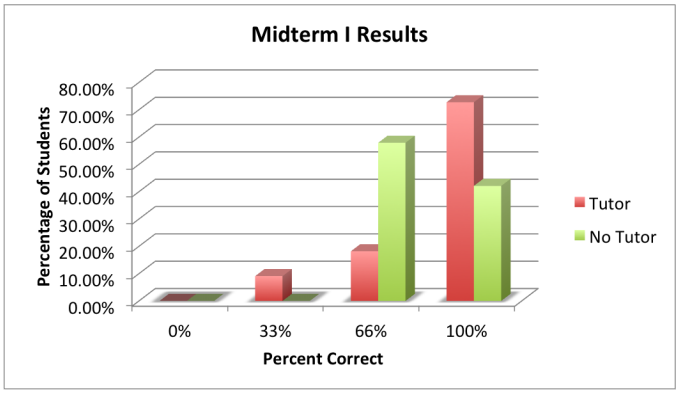

Fig. 1. Midterm score distribution for the three functions questions.

We analyzed the percentage of students in the experimental (tutor) and control (no tutor) groups who answered each specific midterm question correctly. The results are listed in Table III with the p-values from an unpaired two-tailed t-test on each question. Although there are no significant differences, the topic with the largest difference between the two groups explores the caller using a value return by a function with a void return type.

TABLE III
ANALYSIS OF FUNCTIONS MIDTERM QUESTIONS

Topic	Exp.	Control	*p*
Caller tries to use value returned by a function with void return type.	72.73%	47.37%	0.189
Parameter passing and variable scope.	90.91%	100.00%	0.194
Parameter passing by reference.	100.00%	94.74%	0.456

The data from participating students (N=30) was converted to a [0, 1] range for analysis, and statistics are provided in Table IV. We conducted a two-tailed unpaired t-test on the averages [$t(28) = 0.993$, p=0.072] and found no significant difference ($p > 0.05$) in student performance between the experimental (tutor) and control (no tutor) groups.

3) Participation: A post survey was conducted to determine why so few students chose to use the online tutor (N=11). Students were able to select more than one of the listed responses, and an "other" field was provided for students to enter their own reason why they completed (or did not complete) the functions online tutor. The results for why the students completed the functions tutor are displayed in Figure 2. The top reasons were "I said I would" and "I needed the extra points".

TABLE IV
STUDY STATISTICS (FUNCTIONS MODULE)

Group	N	Mean	Std. Dev.	Std. Err Mean
Exp.	11	0.879	0.225	0.068
Control	19	0.807	0.169	0.039

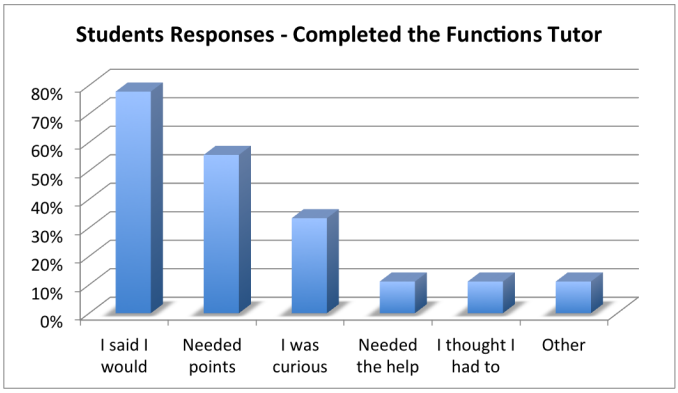

Fig. 2. Post-survey results for why students completed the functions online tutor.

Even though students were reminded, and deadlines were clearly communicated, there were only two reasons stated by students who did not complete the tutor:

- I forgot (100%).
- I wanted to, I just had no time (14%).

The lack of participation led to several refinements in methodology. For the second module (pointers), we made the assignment required, reworked the incentive, and added a paper study guide for those students not assigned to the online programming tutor.

B. Module 2: Pointers

For the pointers module, the experimental group consisted of students who were assigned the Problets tutor on pointers and the control group consisted of students assigned the paper study guide.

1) Pre-test Results: We examined the scores from the students who completed the pre-test (N=55). One student was absent from the pre-test; we feel the omission does not effect the pre-test results. The averages were 44.67% for the experimental (online) group and 39.23% for the control (paper) group. No student received a 100%.

2) Midterm Scores Analysis: We developed four midterm examination questions that focused on the specific topics covered in the *Pointers* tutor; each question was worth 5 points and they contributed to 20% of the overall test score. In the

experimental group, 30 students completed the online tutor; in the control group, 26 students completed the paper study guide.

The average score was 64.17% for those who used the online tutor (experimental) and 70.19% for those who were assigned the paper study guide (control). The distribution for the average scores of the four pointers questions on the second midterm is shown in Figure 3.

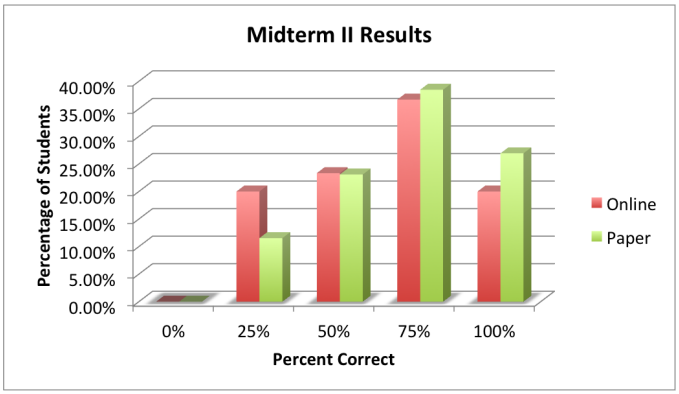

Fig. 3. Midterm score distribution for the four pointers questions.

We also looked at the percentage of students in the experimental (tutor) and control (paper) groups who answered each specific midterm question correctly. The results are provided in Table V along with the p-values from an unpaired two-tailed t-test on each question. Evaluation of the questions separately allows us to see a difference in the memory leak question, where students in the experimental group performed slightly better; however, the difference is considered to be not statistically significant $(p > 0.05)$.

TABLE V
ANALYSIS OF POINTERS MIDTERM QUESTIONS

Topic	Exp.	Control	*p*
Pointer and variable type mismatch.	50.00%	73.08%	0.080
Dangling Pointer.	53.33%	61.54%	0.545
Dereferencing uninitialized pointer.	93.33%	96.15%	0.647
Memory leak.	60.00%	50.00%	0.462

The data from participating students (N=56) was converted to a [0, 1] range for analysis, and statistics are provided in Table VI. We conducted a two-tailed unpaired t-test on the averages [$t(54) = 0.888$, p=0.068] and found no significant difference $(p > 0.05)$ in student performance between the experimental (tutor) and control (paper) groups. Although the difference is not statistically significant, it is interesting that the control group had a lower average on the pre-test but a slightly higher average on the four pointers midterm questions (70.2%) than the experimental group (64.2%).

3) Participation: During our research experiment, we learned the importance of choosing the right incentives to obtain student participation. Making the assignment required

TABLE VI
STUDY STATISTICS (POINTERS MODULE)

Group	N	Mean	Std. Dev.	Std. Err Mean
Exp.	30	0.642	0.260	0.048
Control	26	0.702	0.245	0.048

generated more student participation for the pointers module study than in the first module (functions). With the addition of the paper study guide, all students were given the opportunity for reinforcement and additional practice on key concepts; more than 90% of the students participated.

V. POST ATTITUDINAL SURVEY

Although our primary concern for this study was content knowledge, given the low participation rate for functions, we were interested in student attitudes on their experience. Two weeks after the completion of the study, we deployed a post attitudinal survey with four Likert-style questions. The questions were weighted with *5=Strongly Agree* to *1=Strongly Disagree* and listed in Table VII.

TABLE VII
POST ATTITUDINAL SURVEY QUESTIONS

Please say how much you agree or disagree with the following sentences:

1) I have improved confidence with pointers from participating in this study.
2) I feel I scored better on my exam after having the extra help on pointers.
3) I would use a paper study guide again in the future if given the chance to supplement class instruction.
4) I would use an online tutor again in the future if given the chance to supplement class instruction.

Results from the students who completed the post survey (N=37) are shown in Figure 4 for students who were assigned the online tutor and Figure 5 for students who were assigned the paper study guide. Even though the differences are not statistically significant, the raw data highlights areas that could be explored further.

VI. DISCUSSION

Online programming tutors are designed to allow students to work at their own pace through the use of drill-and-practice problems; they offer the potential benefit to reinforce and/or supplement key concepts being discussed in programming courses. Prior research indicates the use of online-programming tutors are effective in assisting student learning in the introductory programming courses (when measured from pre-test to post-test).

The goal of this study was to determine whether an online tutor would positively impact students' understanding of functions and pointers. In the first module, we determined that there was no significant difference between students who use

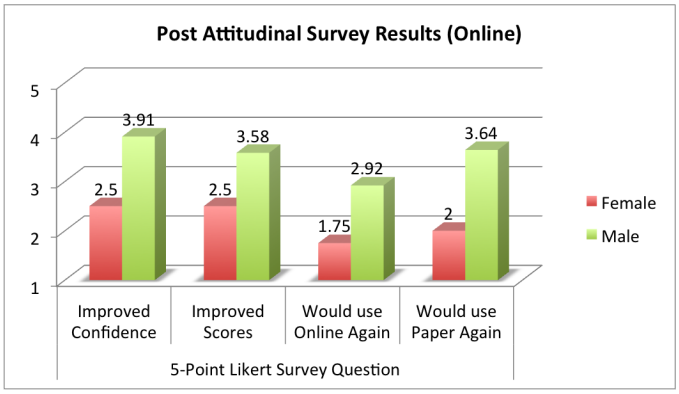

Fig. 4. Post-survey weighted averages from the students in the experimental (online) group.

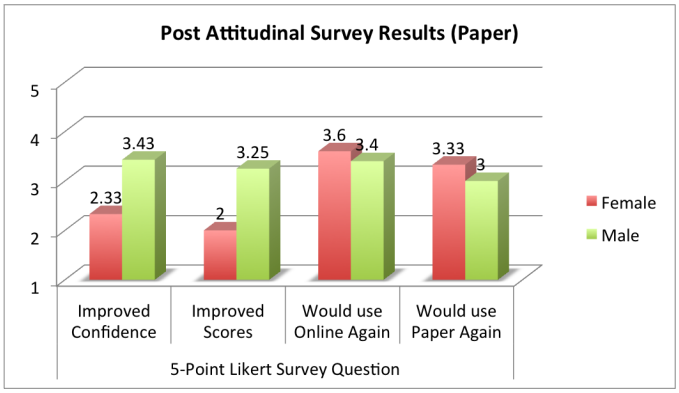

Fig. 5. Post-survey weighted averages from the students in the control (paper) group.

the tutor and the control group who had no extra instruction. The lack of significant effect in this module may be due in part to the small sample size, as it is possible that only the most motivated students spent time with the tutor. The second module compared the performance of students who used the online tutor to students who used a paper study guide. Again we saw no significant difference in performance on the midterm examination questions.

Problets were developed primarily for introductory classes; thus, the material may not exactly match the goals and requirements for those topics within a second semester (i.e., data structures) course. Clearly, if the topics reinforced by the tutor do not correlate exactly with the midterm topics or question phrasing required by the course instructor, this would cause a lack of effect.

For the first module, the majority of students achieved 80% (or higher) on the related midterm examination. Since functions are essentially a review topic for data structures, it is possible that students simply did not need the reinforcement provided by the tutor. The second module compared two interventions (online tutor vs. paper study guide), but there was not a comparison group that received no intervention.

Based on the post survey results, some students showed increased confidence and/or expectation that the additional study would improve test scores. Due to the lack of a comparison group, we don't know whether this was false confidence, or whether they in fact did better than they would have without the additional practice. Our results imply that there is no statistically significant difference based on the type of practice (i.e., paper study guide vs. online tutor). This could be good news for instructors who wish to provide additional opportunity for review, but do not have the time required to create an elaborate tutoring system. More study would be needed to determine if this is in fact the case.

Acknowledgment

We would like to thank Dr. Christopher Painter-Wakefield for his contributions and allowing us to conduct our research study in his data structures course.

References

[1] A. N. Kumar, "Learning programming by solving problems," in *Informatics Curricula and Teaching Methods*. Springer, 2003, pp. 29–39.

[2] "Problets," www.problets.org, accessed: 2015-3-30.

[3] A. N. Kumar, "Results from repeated evaluation of an online tutor on introductory computer science," in *Frontiers in Education Conference (FIE 2011)*. IEEE, 2011, pp. T2H–1–T2H–6.

[4] ——, "The effect of using problem-solving software tutors on the self-confidence of female students," in *ACM SIGCSE Bulletin*, vol. 40, no. 1. ACM, 2008, pp. 523–527.

[5] ——, "Software tutors help female students learn programming concepts just as well as male students," in *World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, vol. 2007, no. 1, 2007, pp. 6840–6845.

[6] ——, "Do female students feel differently than male students about using software tutors?" in *Frontiers in Education Conference (FIE 2006)*. IEEE, 2006, pp. 13–18.

[7] ——, "An evaluation of self-explanation in a programming tutor," in *Intelligent Tutoring Systems*. Springer, 2014, pp. 248–253.

[8] S. Rosenthal and A. N. Kumar, "A rule-based expert system for diagnosing student errors in a tutor on counter-controlled loops," in *AI-supported Education for Computer Science (AIEDCS 2014), ITS*, 2014, pp. 11–20.

[9] A. N. Kumar, "A tutor for using dynamic memory in C++," in *Frontiers in Education Conference (FIE 2002)*, vol. 1. IEEE, 2002, pp. T4G–12–T4G–16.

[10] A. N. Kumar and L. C. Kaczmarczyk, "Programming tutors, practiced concepts, and demographics," in *Frontiers in Education Conference (FIE 2013)*. IEEE, 2013, pp. 773–778.

[11] A. N. Kumar, "Affective learning with online software tutors for programming," in *Psychology of Programming Interest Group (PPIG 2014)*, 2014, pp. 89–98.

[12] R. Lawrence, "Teaching data structures using competitive games," *IEEE Transactions on Education*, vol. 47, no. 4, pp. 459–466, 2004.

[13] G. Gárcia-Mateos and J. L. Fernández-Alemán, "A course on algorithms and data structures using on-line judging," in *ACM SIGCSE Bulletin*, vol. 41, no. 3. ACM, 2009, pp. 45–49.

[14] "Mooshak," mooshak.dcc.fc.up.pt, accessed: 2015-4-8.

[15] S. Patel, "A literature review on tools for learning data structures," 2014.