Incorporating Analogies and Worked Out Examples as Pedagogical Strategies in a Computer Science Tutoring System

Rachel Harsley[1]
rharsl2@uic.edu

Nick Green[1]
ngreen21@uic.edu

Mehrdad Alizadeh[1]
maliza2@uic.edu

Sabita Acharya[1]
sachar4@uic.edu

Davide Fossati[2]
davide@fossati.us

Barbara Di Eugenio[1]
bdieugen@uic.edu

Omar AlZoubi[3]
oalzoubi@cmu.edu

[1]University of Illinois at Chicago, Chicago, IL, United States
[2]Emory University, Atlanta, GA, United States
[3]Carnegie Mellon University in Qatar, Doha, Qatar

ABSTRACT

Analogies and worked out examples are effective means of instruction in a wide variety of learning environments. However, the extent of their effectiveness in Computer Science (CS) education has not been fully explored. We extended our intelligent tutoring system (ITS) for CS data structures, ChiQat-Tutor, to incorporate worked out examples and analogy as teaching strategies. We compare three versions of the system: one that uses standard worked out examples, one that uses analogical worked out examples, and one that uses a pure analogical explanation with separate worked out examples. A study with 66 students showed that students using the standard worked out examples had greater learning gains than students in both analogy conditions. We also found that analogy can be less effective for students with higher prior knowledge. Additionally, we show that some interaction patterns highly correlate with student gains. Overall, the system implementation and results represent a step towards exploring the use of well-established instructional strategies in a computer science ITS.

Keywords

Intelligent Tutoring Systems; Analogy; Worked-Out Examples; Examples; Linked Lists; Computer Science Tutoring

1. INTRODUCTION

As demand for CS professionals rises, so does the importance of teaching computing with engaging, automated, and scalable methods [7]. One such method is the use of intelligent tutoring systems (ITSs). ITSs can provide adaptive feedback to students as they learn a topic. However, other teaching strategies in addition to feedback are possible. The goal of this paper is to analyze the impact of two of such strategies, namely worked out examples and analogy, on the effectiveness of an ITS for CS data structures.

SIGCSE'16, March 2–5, 2016, Memphis, TN, USA.

DOI: http://dx.doi.org/10.1145/2839509.2844637

ITSs provide several benefits to enhancing student learning, including: 1) avoid the one-to-many student teacher model and free teachers to more effectively triage student needs; 2) provide students with individual, customized support tailored to their level of proficiency; 3) allow students to reach common proficiency goals in an efficient and verifiable manner. ChiQat-Tutor aims to provide each of these benefits with the specific focus on CS education. The design of ChiQat-Tutor is informed by a corpus of interaction between students and human tutors. Analysis of tutors' teaching strategies and their effect on learning guide our implementation.

ChiQat-Tutor provides tutoring modules for several CS data structures and algorithms including liked lists, binary search trees, and recursion [5]. The focus of the current study is the linked list data structure module. The system presents a problem to a student in both textual and graphical representation (Figure 1). The student is required to programmatically solve the problem. Moreover, the system provides relevant positive and negative feedback to the student. Example problem types involve linked list node insertion and removal in addition to other more complicated operations.

The paper explores how changes in the instructional design implementation of the tutor affect student learning. Namely we implement and compare the following three strategies in terms of learning gain:

- worked out examples (WOE)
- analogy based worked out examples (AWOE)
- standalone analogy explanations with WOEs (A+WOE)

Moreover, we analyze the factors that contribute to learning gain in each condition and draw general conclusions on the strategies.

2. BACKGROUND

Worked Out Example (WOE): A worked out example is a problem whose solution is given to the student along with the steps for deriving the solution. Examples seem to play a central

role in the early phases of cognitive skill acquisition [9]. Moreover, several studies showed the effectiveness of WOEs for learning in well-structured domains such as physics, programming, and mathematics [9]. Further, WOEs have been implemented in tutoring systems and students achieved deeper conceptual understanding of concepts in comparison to their counterpart without WOEs [8].

In prior work we found moderate but significant correlations between WOE usage by tutors and learning gains in tutees [2]. An earlier experiment with the implementation of WOE in our tutoring system indicated that the intervention did not aid students' learning [3]. One of the possible reasons is that the system did not restrict students to sequential problem solving. Thus, the order in which students were introduced to problems may not have been consistent with the examples and their relevant conceptual principles introduced.

Analogy: Analogies capture parallels across divergent systems. Gentner defines analogies as "partial similarities between different situations that support further inferences" [4]. Analogy's pivotal role in the learning process is well established and some researchers even consider it to be the core of cognition [6]. Gentner shows it can be especially helpful for learners lacking prior knowledge [4].

Our prior work in understanding analogy in CS tutoring dialogues found that the presence of analogical episodes correlated with student learning gains [1]. In our human tutoring corpus, common analogical mappings include comparison of linked lists to movie theatre lines, stacks to cafeteria trays and Legos, and a weaker mapping of binary trees to biological family trees. The tutor tended to initiate the analogical mapping and use it in two ways 1) as an introduction to a particular topic or 2) as a means to extend a problem.

3. SYSTEM DESIGN

ChiQat-Tutor is organized into four primary graphical interface components: 1) Problem View, 2) Feedback View, 3) State Space View, and 4) Command Panel (Figure 1). Unlike a prior study's implementation of worked out examples, we require students to solve problems sequentially [3]. In this manner, students must first successfully solve the foundational problems before being able to advance to more complex problems. Thus, up to seven problems can be unlocked as a student successfully solves problems.

Once a student selects a problem, the linked list's state space representation of the problem is depicted and students can enter code. Each valid line of code modifies the state space. Moreover, our system logs the correctness of code submission as well as the timing interaction influence the feedback of the tutor.

3.1 Worked Out Example (WOE)

The worked out example condition allows students to open an example that is relevant to the problem they are attempting to solve. The request for example button can be clicked at any time while a student attempts to solve a problem. Each problem has exactly one example. Students click to progress through the example, however, they may exit the example at any time by reselecting a problem.

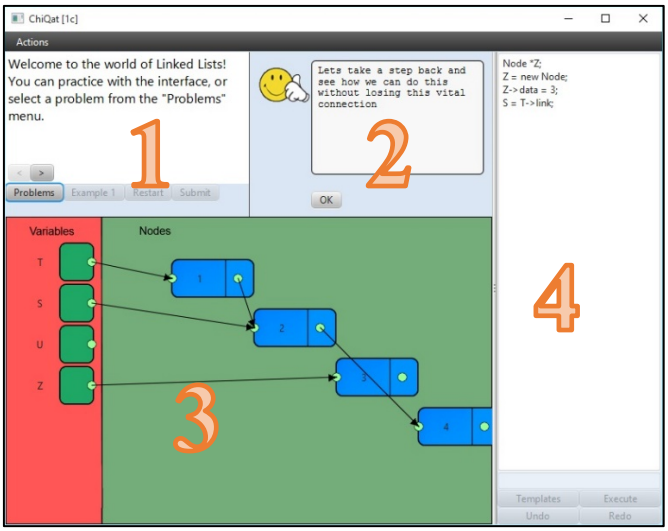

Figure 1. The GUI is split into four parts 1) Problem View 2) Feedback View 3) State Space View, and 4) Command Panel.

We designed the examples so that the principles needed to solve the related problem are illustrated in the example. It is then left to the student to perform the knowledge transfer to their specific problem and hopefully generalize the principles at large.

3.2 Analogical Worked Out Example (AWOE)

In the case of an analogical worked out example, the structure of interaction with the example is identical to the standard worked out example. However, the content of the example is based on an analogy as opposed to the pure linked list data structure. Given prior work analyzing a human CS tutoring corpus, the chosen analogy maps linked lists to a movie theatre line. A step of the analogical worked out example is depicted in Figure 3.

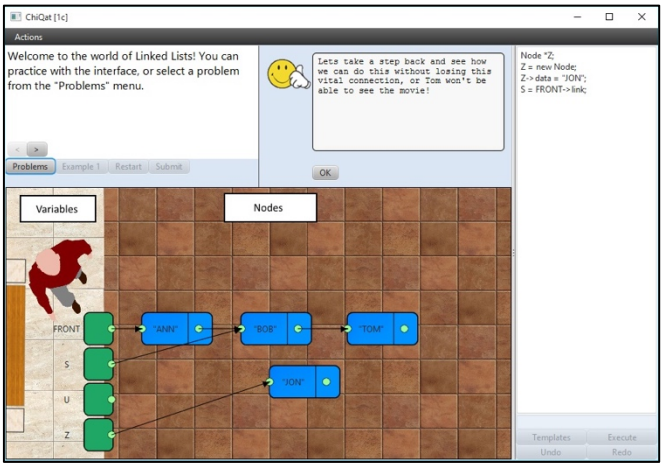

Figure 3. The image depicts another example of inserting a node into a list.

3.3 Analogical Explanation with Worked Out Example (A+WOE)

The second approach to integrate analogy is based on the fact that analogy is most likely to appear at the beginning of a human tutoring session [1]. Thus, when the linked list tutorial begins, the system displays a window describing an analogy of people standing in a line as depicted in Figure 5 (full analogy not shown).

In this condition, students are also able to explore the same WOEs used in the WOE condition.

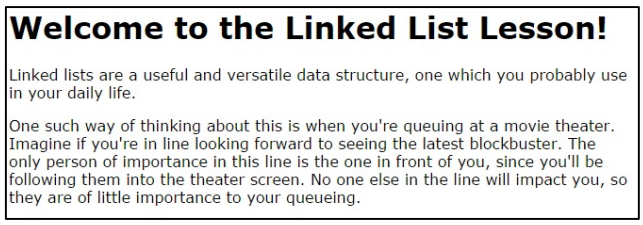

Figure 5. Portion of the analogy presented in a popup at the onset of the A+WOE condition.

4. EXPERIMENT

We conducted the experiment in a 2nd year CS programming course. The course focuses on software development tools and practices, and on the implementation of foundational data structures and algorithms such as stacks, linked list, and depth first search to solve real-world problems. For example, students are given a project to implement a linked list in C++ with support for insert, delete, and find operations. The following week they are required to implement a program which uses the linked list to store a waiting list for a restaurant.

Our experiments ran over four sections of the course, all taught by the same instructor and composed of a comparable mix of students. Students provided informed consent for the experiment. The students took a pre-test for 10 minutes; then they worked with ChiQat-Tutor for 40 minutes; finally they took a post-test (identical to the pre-test). Students' interaction with the system was continuously logged. There were three conditions with a total of 66 users; 23 (WOE), 21 (AWOE), and 22 (A+WOE).

Tests were graded between three independent graders, and their median score was selected. Both pre and post tests are identical and derived from prior work analyzing human CS tutoring dialogues. The first problem depicts the graphical state of a linked list. It is accompanied by code. Students are asked to describe visually and/or with words the state of the list after the accompanying code is executed. The second problem asks students to find the error in an almost exact replication of the code in problem one save for a semantically devised error. The third and final problem requires students to move the first node of the list depicted in problem one to the end of the list. We use use the following measure of learning gain to assist in our analysis of learning: *gain =post_test_score - pre_test_score*.

5. DISCUSSION/RESULTS

Of foremost importance in evaluating the system is the answer to the question of whether or not students have learned. Subsequent to this, is the question to what extent have they learned and finally what factors contributed to the learning. In answer to the primary questions, the students do learn in each of the conditions. The WOE condition achieves learning gains similar to those of the human tutoring corpus. The AWOE and the A+WOE seem to be lower in learning gain. However, ANOVA did not reveal an overall significant difference among the five groups when computing standard learning gain.

With the premise established that the system is effective in helping students learn, we aimed to further our analysis in terms of how learning occurs. In order to more deeply understand the features of the student interaction with the system, we exhaustively logged interactions. From the log files, several features have been analyzed using linear regression with each variable being used as a predictor of learning gain in several independent models.

Table 1. Learning gain of students in five conditions.

Tutor	N	Pre-test		Post-test		Gain	
		μ	σ	μ	σ	μ	σ
Human	54	.40	.26	.54	.26	.14	.25
ChiQat WOE	23	.41	.18	.55	.22	.14	.17
ChiQat AWOE	21	.50	.17	.61	.15	.11	.18
ChiQat A+WOE	22	.52	.21	.60	.23	.09	.21

Table 2. Features used in multiple linear regression models

Feature	**Description**
Collection_Duration	Time in milliseconds that the user used the system (from login to logout)
Example_Completed	User completed viewing the example
Example_Requested	User requested an example
Example_Started	An example has started to be played
Executing_ Example_Step	User clicked the 'next' button, moving an example to the next step
Feedback_Given	Feedback has been given to the user
Final_Pre	Pre-test score
Node_Clicked	User clicked on a node within the graphical problem solving area
Operation_Execution _Submitted	User submitted an operation, such as a line of code, for a problem
Operation_Redone	User elects to redo a step that was previously undone
Operation_Undone	User elects to undo a previously submitted step
Question_Answer_ Response	The system responded to an answer that the user has given to a posed question
Question_Given	A question has been posed to the user from the system
Solution_Submitted	A solution has been submitted for validation by the user to the system
Solved_Problems	Total number of problems the user solved
Template_Help_ Item_Selected	User clicked on a code template item in order to help them write code
Template_Help_ Requested	User requested to see all programming commands available in the system
Tutorial_Duration	Total duration that the user viewed a lesson tutorial
User_Acknowledge	User clicks the 'ok' button, signifying that they have acknowledged any feedback

The number of measured features (n=39) exceeds the number of examples per condition. Thus, we created a regression tree in order to determine important features in regards to learning gain.

From this point, we derived a linear additive model using the largest possible subset of important features capable of being added to the model. Given the result of the multiple regression analysis, we then found the minimum adequate model. This model is a reduction of the first model which aims to only keep features that are significant. Thus we show and discuss the statistically significant multiple linear regression model that selects a set of features resulting in the highest adjusted R^2. Of the initial 39 features, 27 were actually incorporated in the most explanatory models across the three conditions. The features and their descriptions are given in Table 2.

Table 3 and Table 4 show the statistically significant models we obtained as described above. ß are the best estimates of the coefficients of each and p shows the level of significance. R^2 explains how well features can describe variations in learning gain and it is given as well.

Table 3. **Model for the WOE condition learning gain.**

	ß	t value	p
(Intercept)	-9.816	-7.733	0.005 **
Feedback_Given	-.189	-2.150	.121
Collection_Duration	.002	2.662	0.07622 .
Operation_Undone	-.068	-1.342	.272
User_Acknowledge	.286	5.003	0.015 *
Executing_Example_Step	-.185	-5.195	0.014 *
Operation_Execution_Submitted	.111	4.664	0.019 *
New_Problem_Selected	.247	1.094	.354
Example_Started	-3.407	-2.106	.126
Example_Completed	2.644	1.568	.215
Example_Requested	1.357	2.216	.113
Problem_List_Requested	.247	2.100	.127
Final_Pre	-.111	-1.154	.332
Template_Help_Requested	.079	4.204	0.025 *
Lesson_Started	2.518	3.515	0.040 *
Node_Clicked	.060	7.276	0.005 **
Operation_Redone	-.567	-4.900	0.016*
Problem_Restarted	.326	1.749	.179
Question_Answer_Response	-.285	-1.770	.175
Solution_Submitted	-.276	-8.114	0.003 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Multiple R-squared: 0.9943, Adjusted R-squared: 0.9579

To our surprise, in the case of the worked out example condition, executing an example step is significantly negatively correlated to learning gain (p<.01). However, completing an example is positively correlated with a much larger coefficient. This seeming oxymoron may suggest that student use of the example signifies their lack of understanding of a particular principle. Thus, the further students explore an example, it may signify the depth of their uncertainty of a concept. This is also suggested by the much larger by the much larger positive correlation between example completion and learning gain. In a sense, if a student begins the example with uncertainty and completes the example, they learn the concept in question. However, if a student advances in examples without completing it, they may not attribute the same learning gain.

In general, submitting an operation for execution is significantly positively correlated with learning gain (p<.01). We also note that this submission does not account for whether the executed code is correct (syntactically or otherwise). Thus, in general, the more students attempt to find a correct solution, the more they learn. This is not the case for submitting a solution. Submission signifies that the student believes the entire problem has been solved. In fact, submitting entire solutions is strongly negatively correlated with learning gain (p<.001). This connection is clear because students with multiple solution submissions for the same problem are inherently missing some level of conceptual knowledge needed to solve the problem through they believe their work is viable to submit.

Also of interest is node clicks in the graphical representation of the list. This feature is strongly significantly correlated to learning gain (p<.001). It is not difficult to imagine that as students reason about either a worked out example or their own code, they rearrange the linked list nodes in ways that make sense for their understanding. This may help them internalize the conceptual principles. The high correlation of this feature serves as a testament to the power of the system to graphically represent the often difficult and abstract concept of linked lists.

The redo operation is significantly negatively correlated to student learning in the WOE condition (p<.01). The redo exemplifies student uncertainty about a concept. The student submits and operation, undoes the operation, then redoes the same operation.

Table 4. **Model for the AWOE condition learning gain.**

	ß	t value	p
(Intercept)	13.456	5.612	0.001 **
Final_Pre	-.947	-4.180	0.006 **
Template_Help_Requested	-.175	-1.968	0.097 .
Node_Clicked	.019	1.481	.189
Collection_Duration	-.004	-3.333	0.016 *
Template_Help_Item_Selected	.103	.944	.381
Starting_Lesson_Tutorial	-7.456	-1.686	.143
Tutorial_Duration	.107	2.729	0.034 *
Operation_Redone	.371	1.257	.255
Question_Given	1.242	1.904	.106
Operation_Undone	.189	3.869	0.008 **
Feedback_Given	.152	2.318	0.060 .
User_Acknowledge	-.139	-1.862	.112
Question_Answer_Response	-1.199	-1.839	.116
Executing_Example_Step	-.129	-2.364	0.056 .

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Multiple R-squared: 0.9434, Adjusted R-squared: 0.8113

Finally, in the WOE condition it is worth noting that the learning gain and pre-test score are not significantly correlated. This

finding is important because it signifies that the system is able to influence learning gain irrespective of the prior knowledge of the student.

The regression analysis reveals a significant negative correlation between pre-test score and learning gain in the AWOE condition. Though the mean pre-test score for the AWOE (μ=.50) is slightly higher than the pre-test scores for the WOE condition ((μ=.41), the difference is not significant. This suggests that the AWOE attributes a ceiling effect on learning gain. This effect is explored further on in this section.

In the AWOE condition, requesting the template help is significantly negatively correlated to learning gain ($p<.05$). This correlation highlights that students are not internalizing the syntactic rules and are instead uncertain of what to code and the proper notation. They instead rely heavily on the system for assistance. This is further demonstrated by the difference in coefficients and significance of Template_Help_Item_Selected versus Template_Help_Requested. In essence, some students request template help much more often than they select a template from those provided. They attempt to use the templates as kick-starter for understanding but eventually end up performing multiple requests for templates and finally selecting a template.

An example of this pattern of interaction is given in Table 5. In this case, the student starts a lesson and immediately requests template help. The student again requests template help and then submits their code with a clear syntax error. After receiving feedback, the student requests further help and submits code, this time successfully. The student continues this cycle of requesting help one or more times, selecting a template, then submitting an operation throughout the lesson.

Table 5. Sample log of a student using "Template Help Requested" and "Template Help Item Selected"

Time	Action	Argument
15:49:54	Template Help Requested	
15:50:17	Template Help Requested	
15:50:18	Template Help Item Selected	? = new Node;
15:50:20	Operation Execution Submitted	3 = new Node;
15:50:21	Feedback Given	*omitted*
15:50:31	User Acknowledge	
15:50:37	Template Help Requested	
15:50:38	Template Help Item Selected	Node *?;
15:50:44	Operation Execution Submitted	Node *temp;

Further, undoing an operation attributes a significantly positive correlation to learning gain ($p<.001$). This feature allows students to remove a previously submitted operation. An undo represents a student's change in approach in regards to the problem. This may be representative of a student's newly found, deeper understanding of the problem goal and possible errors in their logic. Note, the undo cannot be a student attempting to correct syntactic errors as syntactic errors do not show up in the command panel as submitted code. Thus, we hypothesize the student may be attempting to solve higher order, semantic issues when an undue transpires.

In the final condition of A+WOE, the R^2 value is the lowest among the conditions (Multiple R-squared: 0.9827, Adjusted R-squared: 0.6376). Moreover, the model contains no significant predictors of learning gain. The model is not shown.

Table 6. Comparative overview of select features used in multiple linear regression models along with their role in each best fit regression model. (+) denotes positive correlation (-) denotes negative correlation. Significant correlations marked.

Feature	Learning Gain		
	WOE	AWOE	A+WOE
Collection_Duration	+ <.05	- <.01	+
Example_Completed	+		-
Example_Requested	+		
Example_Started	-		-
Executing_Example_Step	- <.01	- <.05	+
Feedback_Given	-	+ <.05	-
Final_Pre	-	- <.001	+
Node_Clicked	+ <.001	+	+
Operation_Execution_Submitted	+ <.01		-
Operation_Redone	- <.01	+	
Operation_Undone	-	+ <.001	+
Question_Answer_Response	-	-	-
Question_Given		+	+
Solution_Submitted	-		
Solved_Problems			+
Template_Help_Item_Selected		+	
Template_Help_Requested	+ <.01	- <.01	-
Tutorial_Duration		- <.05	
User_Acknowledge	+ <.01	-	+

Table 6 aims to provide a comparative look at the three conditions in terms of the factors that contribute to learning. A summary of findings follows:

- **Time on Task:** Interestingly, the amount of time spent with the system is significantly positively correlated with the WOE condition ($p<.05$), however, it is significantly negatively correlated with the AWOE condition ($p<.01$).
- **Examples:** Both the WOE and AWOE conditions show that executing example steps themselves significantly negatively correlate to learning.
- **Tutor Feedback**: In the AWOE condition, the volume of tutor feedback is significantly positively correlated to learning ($p<.05$). However, tutor feedback in the other conditions is negatively correlated to learning.
- **Prior Knowledge**: The pre-test score is very strongly significantly correlated to learning in the AWOE condition ($p<.001$). However, prior knowledge of the student in both the WOE and A+WOE conditions do not play a significant factor.
- **Graphical List Interaction**: In all conditions, interaction with linked list nodes is positively correlated with learning. In the WOE case the correlations is quite significant ($p<.001$).
- **Undo/Redo**: The undo behavior is positively correlated to student learning in both analogy conditions with a significant correlation in the AWOE condition ($p<.001$). However, the undo behavior is negatively correlated to learning for the WOE condition. Similarly, the redo behavior is significantly negatively correlated to learning in the WOE condition. Yet, in the AWOE condition it is positively correlated.

The strong negative correlation between the pre-test score and learning gain along with the conditional inference tree of both analogy conditions revealed an interesting split in users based on learning gain. The split occurred by dividing the users into those who scored <=5 (out of 15 max) on the pre-test and those who did not. A similar split occurred using the score cutoff of 7. We ran unpaired t-test to determine if the difference in learning gain was significant. The difference in learning gain was significant in each case ($p=0.0006$ and $p= 0.0281$ respectively). In comparison, the WOE condition did not have significant difference in learning gain with either threshold.

To make sense of this split, we performed t-test to compare the low pre-test analogy conditions with the low pretest WOE condition. We also conducted the same comparison for the high pre-test groups. The comparison of learning gains for WOE pre-test scores >5 and the combination of both analogy conditions pre-test scores >5 did not yield a significant difference ($p= 0.1137$). However, the threshold change of pre-test scores >7 did yield a significant difference ($p= 0.0310$). The significant difference suggests that the analogy conditions were least beneficial, and in fact damaging to students with higher prior knowledge in comparison to the standard WOE condition. Note, there was no significant difference in the low to low group comparisons.

6. CURRENT WORK/CONCLUSIONS

Work is ongoing to improve the ITS and ensure it incorporates the most effective strategies for helping students to learn CS. Our focus on foundational CS data structures and algorithms allows us to explore unchartered territories in the world of tutoring systems. Thus, the effectiveness of some common instructional designs seen in other fields have yet to be tested in our domain. In this study we expanded on prior work analyzing the use of analogy and worked out examples used by human tutors for CS. We integrated these strategies into our tutoring system in hope of 1) improving student learning gain and 2) understanding the factors that contribute to learning in each condition.

We explored three conditions: 1) worked out examples, 2) analogical worked out examples, and 3) standalone analogical explanations with worked out examples. To the benefit of students, our study shows that the system contributes to learning gains in all three conditions. However, the WOE condition exceeded the learning gain of the other two conditions and achieved gains comparable to a human tutor as well as prior best implementations of the tutoring system.

In general, we were able to create highly explanatory models of the three conditions. However, we caution the potential for the high number of factors and multiple comparisons to yield false positives in significance testing. With this in mind, we plan to evaluate individual factor manipulation for a more accurate understanding of the features relationship to learning gains.

The study showed that advancing through example steps in itself is not directly correlated to learning gains. However, completing the example is effective. In the case of the WOE, the more code students submit, regardless of correctness, the more they learn. Further, the study showed strong positive correlations between student exploration of the graphical linked list representation and learning across each condition. The analysis of this feature showcases the power of an ITS to adapt to individual users in regards to the code they submit, and their overall interaction with the problem.

Finally, the study shows that analogy is least beneficial to students with higher prior knowledge. However, students in the analogy condition attribute undo behavior that alludes to deeper understanding of concepts and correlates to learning gains. We are currently analyzing the abundance of log data to understand the intricacies of this behavior. Additionally, we are performing a finer grained analysis of analogy in regards to its benefits for particular concepts which can be extracted from each of the seven problems. A future implementation may prove analogy to be beneficial in certain cases.

7. ACKNOWLEDGEMENTS

This publication was made possible by NPRP grant 5-939-1-155 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

8. REFERENCES

[1] Alizadeh, M., Di Eugenio, B., Harsley, R., Green, N., Fossati, D. and AlZoubi, O. 2015. A Study of Analogy in Computer Science Tutorial Dialogues. *Proceedings of the 7th International Conference on Computer Supported Education (CSEDU 2015)* (2015).

[2] Di Eugenio, B., Chen, L., Green, N., Fossati, D. and AlZoubi, O. 2013. Worked Out Examples in Computer Science Tutoring. *Artificial Intelligence in Education*. H.C. Lane, K. Yacef, J. Mostow, and P. Pavlik, eds. Springer Berlin Heidelberg. 852–855.

[3] Di Eugenio, B., Green, N., Alizadeh, M., Harsley, R. and Fossati, D. 2015. Worked-out Examples in a Computer Science Intelligent Tutoring System. *Proceedings of the 16th annual ACM SIGITE conference on Information technology education* (Chicago, IL, USA, Oct. 2015).

[4] Gentner, D. 1998. Analogy. *A companion to cognitive science*. (1998), 107–113.

[5] Green, N., AlZoubi, O., Alizadeh, M., Di Eugenio, B., Fossati, D. and Harsley, R. 2015. A Scalable Intelligent Tutoring System Framework for Computer Science Education. *Proceedings of the 7th International Conference on Computer Supported Education (CSEDU 2015)* (2015).

[6] Hofstadter, D.R. 2001. Analogy as the core of cognition. *The analogical mind: Perspectives from cognitive science*. (2001), 499–538.

[7] Monge, A.E., Fadjo, C.L., Quinn, B.A. and Barker, L.J. 2015. EngageCSEdu: Engaging and Retaining CS1 and CS2 Students. *ACM Inroads*. 6, 1 (Feb. 2015), 6–11.

[8] Schwonke, R., Renkl, A., Krieg, C., Wittwer, J., Aleven, V. and Salden, R. 2009. The worked-example effect: Not an artefact of lousy control conditions. *Computers in Human Behavior*. 25, 2 (Mar. 2009), 258–266.

[9] VanLehn, K. 1996. Cognitive Skill Acquisition. *Annual Review of Psychology*. 47, 1 (1996), 513–539.