STUDENT BEHAVIOR WITH WORKED-OUT EXAMPLES IN A COMPUTER SCIENCE INTELLIGENT TUTORING SYSTEM

Nick Green
University of Illinois at Chicago
Chicago, IL, USA

Davide Fossati
Carnegie Mellon University in Qatar
Doha, Qatar

Barbara Di Eugenio
University of Illinois at Chicago
Chicago, IL, USA

Rachel Harsley
University of Illinois at Chicago
Chicago, IL, USA

Omar AlZoubi
Carnegie Mellon University in Qatar
Doha, Qatar

Mehrdad Alizadeh
University of Illinois at Chicago
Chicago, IL, USA

ABSTRACT

The computing industry is currently facing a huge deficit in talent entering the industry. Even though enrollment in Computer Science (CS) degrees is climbing, many students drop out due to hurdles in grasping fundamental concepts. ChiQat-Tutor, our new CS Intelligent Tutoring System (ITS) helps students overcome difficulties by teaching CS concepts such as data structures. Content aside, an important aspect is how this material is taught. Here we show our work on using Worked-out Examples (WOE) in the linked list tutorial of our ITS. Contrary to previous literature, we found that WOEs are not a silver bullet, and may not be an effective teaching strategy for some students.

KEYWORDS

Computer Science Education, Intelligent Tutoring Systems, Linked Lists, Worked-out Examples.

1. INTRODUCTION

Starting a career in computing is not easy. Fundamental concepts are difficult to grasp and require new ways of thinking. Unfortunately, this is having an impact on the industry with high levels of attrition in Computer Science (CS) courses (Lewis, 2010). High quality and readily available education may help increase retention, however, this may not always be possible. Such education can be costly, as well as inaccessible to the majority of society, especially in underprivileged communities. Intelligent Tutoring Systems (ITS) may help provide a viable alternative.

We are developing a new ITS, ChiQat-Tutor (Green et al., 2015), specifically for the CS domain. Scalability has been included at its core, with the possibility of adding new lessons and teaching strategies. In order to develop teaching strategies, we turned to a corpus of human tutoring data which we collected and annotated (Di Eugenio, Fossati, Ohlsson, & Cosejo, 2009). Strategies our tutors use include different types of feedback (already modeled and deployed), Worked-out Examples (WOEs) and analogy.

In this paper, we investigate the effectiveness of including the WOE teaching strategy into our tutoring system. The investigation centers around the linked list data structure (a fundamental concept in CS) tutorial within the tutoring system, which we deployed in undergraduate Computer Science lab sessions.

2. RELATED WORK

WOEs are a strategy that teaches by example (Sweller, 2006) (Atkinson, Derry, Renkl, & Wortham, 2000). This strategy was shown to induce greater learning gains for some students, mostly on algorithmic conceptual topics (Renkl, 2005). An explanation for the observed greater learning in novices is based on cognitive load theory (Sweller & Cooper, 1985). Sweller suggests that initial learning should focus on building schemas, which are the fundamentals of identifying similar situations and states from where problems can be easily and rationally solved. It is thought that traditional problem solving causes issues in schema development due to the additional cognitive load on working memory that is induced in the problem solving activity (Miller, 1956). WOEs are useful in that they allow learners to analyze the solution to a problem without having this extra cognitive overhead. However, WOEs may not be effective in all cases, for example, individuals with greater knowledge of a topic may make greater gains from problem solving rather than by studying examples. A WOE can be broken into three distinct stages: problem formulation; solution steps; and final solution.

Table 1. Worked-out example for a binary search tree problem

let’s come here and say we searched for 9
um we’d come down here we’d check 5
9 is greater than 5 so we go to its right child
um 9 is greater than 7 we’d check its right child
9 is greater than 8 and we check its right child
9 is equal to 9, would return it all the way up to 5

An example of a WOE can be seen in Table 1. In prior work (Di Eugenio et al., 2009), we recorded (audio and video) 54 human-human tutoring sessions on CS data structure fundamentals. This data was transcribed and annotated for useful features. From these dialogues, we found moderate but significant correlations between WOEs as used by tutors and learning gains in CS tutoring (Di Eugenio, Chen, Green, Fossati, & AlZoubi, 2013). Others have also been able to deploy WOEs in ITS’, usually with positive results. (Najar & Mitrovic, 2013) utilized WOEs in their CS ITS for teaching SQL queries. They concluded that while useful and promoted learning, it was best to use with tutored problem solving. (McLaren, Lim, & Koedinger, 2008) integrated WOEs into an ITS that also includes tutored problem solving. No significant learning gains were observed over problems only, students were able to learn faster.

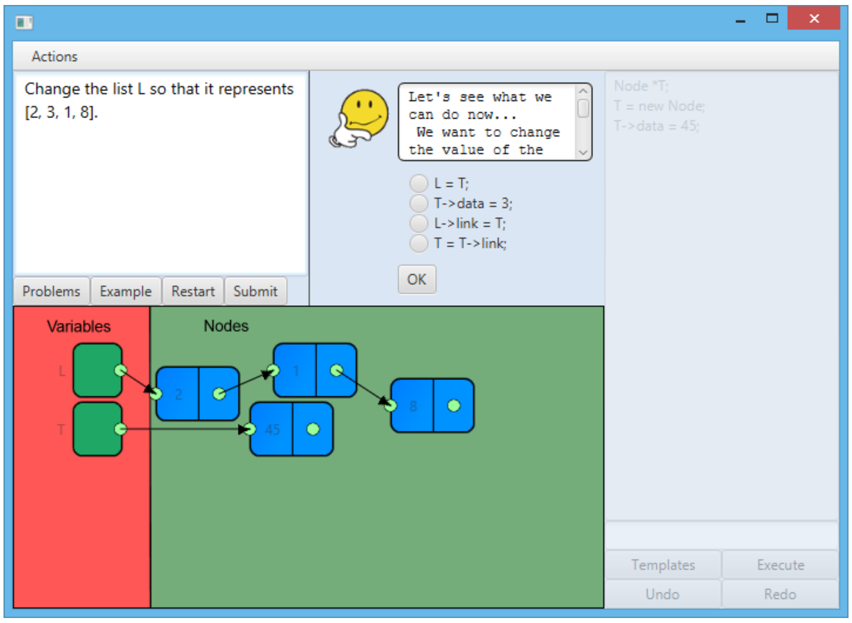
3. THE CHIQAT-TUTOR SYSTEM

3.1 Application

ChiQat-Tutor (Figure 1) is our new ITS to support CS education. The linked list module of ChiQat-Tutor has been ported from iList, a previously tested and deployed ITS for linked lists (Fossati, Di Eugenio, Ohlsson, Brown, & Chen, 2015). The prior system used various feedback mechanisms to aid students in understanding linked lists via problem solving only. ChiQat-Tutor builds upon these successes by using a novel, scalable framework whereby new lessons, teaching strategies, and utilities, can be developed and deployed independently, however, they can interact with each other; hence a strategy can be utilized in multiple lessons.

The linked list lesson includes the same seven problems as our prior system. Each problem exercises different properties of the linked list data structure. For example, problem 1 requires the student to insert a node into a list, while problem 3 needs a node to be removed from one. Each problem can be solved by the student entering C++ or Java commands to manipulate the list, which can be visualized in the GUI. Once students are confident on the overall solution, they can click the ‘Submit’ button to automatically check the validity of the solution, whereby they will be told if it is correct or not.

The ITS records the interaction with the student in the form of log messages. There are approximately 30 types of events, such as starting a new problem; performing a solution operation; and requesting an example.


Change the list L so that it represents [2, 3, 1, 8].

```
Let's see what we
can do now...
 We want to change
the value of the
```

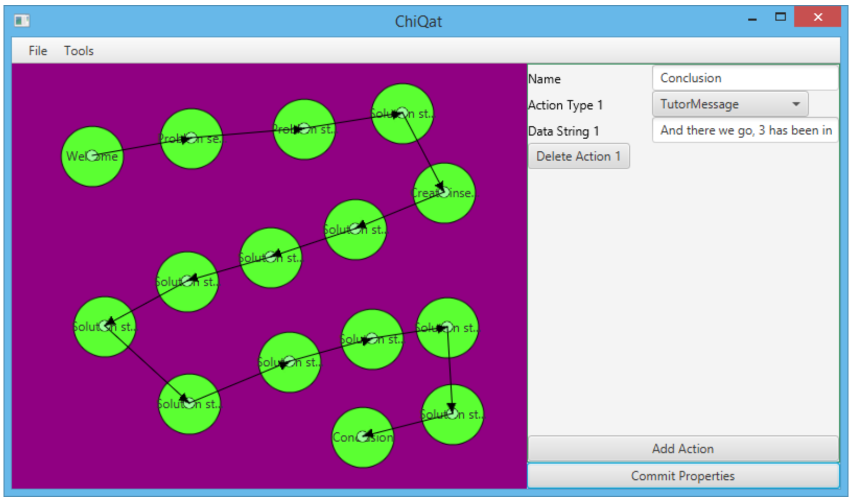
- L = T;
- T->data = 3;
- L->link = T;
- T = T->link;

```
Node *T;
T = new Node;
T->data = 45;
```

Figure 1: Linked list lesson in ChiQat-Tutor

3.2 Worked-out Examples in ChiQat-Tutor

We created a dedicated worked-out example module that encapsulates the entire functionality of the teaching strategy, including the operation of the strategy and authoring of examples. The WOE module includes an example editor (Figure 2) and an engine whereby the example can be played out. WOEs are executed by a student by clicking an 'Example' button, where a relevant WOE for the current problem is played back.



Name	Conclusion
Action Type 1	TutorMessage
Data String 1	And there we go, 3 has been in

Figure 2: Worked-out example editor

Worked-out Examples in this system are contextual, where there is a relevant example for each of the seven problems available in the system. An example is played back to the student once they click the 'Example' button during a problem. Upon execution, the intelligent tutor will begin a step-by-step playback of the example. Each step consists of either dialogue from the tutor, the tutor performing an operation on the linked list, or visually indicating points of interest in the visual representation of the linked list.

The WOEs in this system offer a minimum degree of interactivity. Once started, users can proceed to the next step of the example by clicking the 'OK' button in the tutor window, or they can exit the example by selecting another problem via the 'Problems' button.

4. METHODS AND PARTICIPANTS

We evaluated the system by deploying it to our target audience (novice CS students) under experimental conditions, whereby a subset of users had the WOE feature enabled, whilst the rest did not. The ITS recorded detailed logs of user actions and events, such as the number of problems solved and how many times they manipulated the GUI.

We ran two separate experiments five months apart, both of which were conducted over four contiguous lab sessions as part of a second year course in CS data structures. Both experiments involved the same course in subsequent semesters, with the same instructor and similar student population, at an equivalent point in the semester. Students were given 40 minutes to use the linked list tutorial, which included all seven problems. Pre/post tests of 10 minutes were also administered. In the first set of experiments, from September 2014 (Sep14), there were two conditions; half of the students (43) had WOEs available via the 'Examples' button, while the remaining (39) did not. The second round of experiments were conducted in February 2015 (Feb15) with three different conditions, one of which was the same WOE condition (23) as in the prior round of experimentation. This round did not include a No WOE condition. Tests were randomized and anonymized, then graded between 3 independent graders.

5. RESULTS AND ANALYSIS

Contrary to our initial hypothesis, students *do not* appear to automatically benefit from having WOE access in our ITS. The most striking observation is that there was a high degree of inconsistency between the two WOE conditions.

Table 2 shows the learning gain (as calculated by gain = post-test - pre-test) for each of the tested conditions, as well as with using a human tutor and no intervention at all. Firstly, the Sep14 experiments show a drastic difference in performance between the two groups whereby the group without WOEs performed close to that of being given a human tutor, while the WOE condition made no significant gains at all. This in itself would suggest that either WOEs are not effective in our particular ITS, or perhaps, the WOEs in the ITS were badly designed and inhibited student growth.

Table 2. Learning gains of students

Tutor	**N**	**Pre-test**		**Post-test**		**Gain**	
		μ	**σ**	**μ**	**σ**	**μ**	**σ**
No tutor (control)	53	.34	.22	.35	.23	.01	.15
ChiQat-Tutor without WOE (Sep14)	39	.45	.14	.56	.22	.11	.25
ChiQat-Tutor with WOE (Sep14)	43	.50	.22	.48	.25	-.01	.23
ChiQat-Tutor with WOE (Feb15)	23	.41	.19	.56	.22	.14	.17
Human	54	.40	.26	.54	.26	.14	.25

However, the gains made from the second round of experiments in Feb15 differ from the prior round by yielding the same learning gains as that from a human tutor. This suggests that the WOEs used in the system do not strictly inhibit growth in the vast majority of students, although there may be differences in the two groups that made the use of WOEs effective or ineffective. Figures 3a and 3b show the breakdown of learning gain between the two WOE conditions. These show a striking difference between the benefits gained in the two conditions with the Feb15 condition scoring consistently higher than in Sep14. It is interesting to see that the Sep14 condition shows far more variation with no clear common score, whilst the Feb15 condition has half of the students either gaining in the region of .25 or -.05.

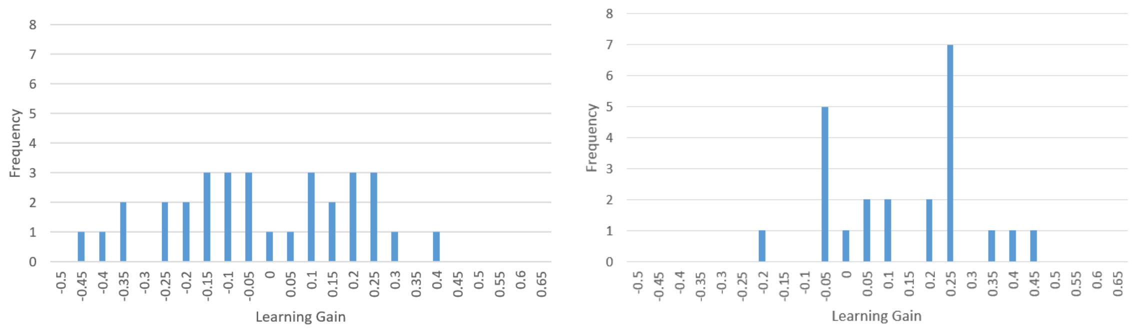


Figure 3: (a) Gain for Sep14 WOE (b) Gain for Feb15 WOE

5.1 Log File Analysis

ChiQat-Tutor logs a variety of events that we can use to get a better understanding of what occurred during the experiments. The logs are primarily composed of: time stamp (in milliseconds); event type; and user ID. We cleaned and processed the logs, then we extracted several features to investigate the interaction of students with our system. Figure 4 gives some insight into some differences between the groups on a problem by problem basis. Firstly, Figure 4a shows the problem success rate for each of the three conditions. The Sep14 WOE condition performed the worst out of all conditions with a lower success rate for all problems. The Feb15 WOE condition does show a greater success rate initially, but under performs from problem 4 on to the Sep14 non-WOE condition. This may be due to the additional time consumed by the students while studying examples, not giving them as much time to progress on more advanced problems. Figure 4b shows the amount of time students spent on each problem, without taking the time spent studying examples into account. All the conditions show that students become more efficient at problem solving as they work through the problems. Interestingly it does not appear that viewing examples reduced the amount of time to solve problems.

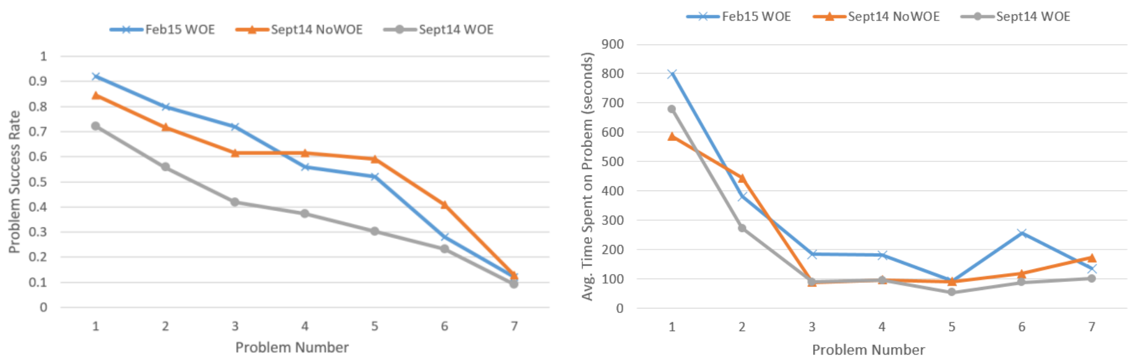


Figure 4: (a) Success solving problems (b) Time spent on problems

5.2 Worked-out Example Behavior

Figure 5 shows two diagrams indicating how students used WOEs over all seven problems. Figure 5a shows a breakdown of WOE usage per problem which in both conditions shows a decline of example usage as problems progress. However, there is a large increase of example usage on problems 6 and 7. This makes sense as problems 6 and 7 differ significantly from the prior problems as they require the student to write whole blocks of code to manipulate a list rather than line-by-line. Also, the latter problems do not include as sophisticated feedback as the prior problems do. Interestingly, there is a very large difference between the reduction rate over problems in both conditions, whereby the Feb15 group appears to utilize the feature far more than the

Sep14 group after the first problem. This may indicate that higher WOE usage in subsequent problems may increase learning gains. The time spent using WOEs (Figure 5b) further reinforced the idea that the Feb15 students utilize the feature more-so than Sep14 students by showing that students tend to, on average, spend more time studying WOEs. The similarities here indicate that while more students use WOEs they spend a similar amount of time studying them.

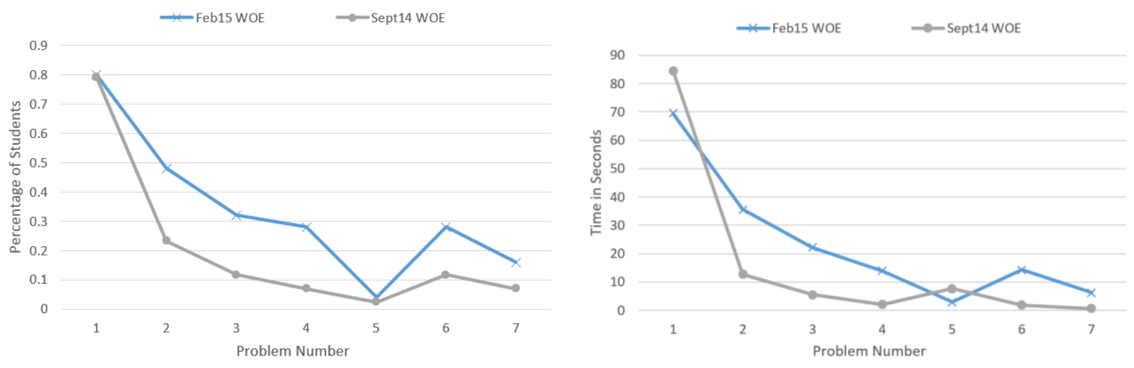


Figure 5: (a) Students using WOEs   (b) Time spent studying WOEs

Next we took a more in-depth look at student behavior when using examples. Due to the high uptake of WOEs on problem 1, and then the rapid degrading of usage on subsequent problems (for Sep14 at least), we chose to focus on examples used exclusively in problem 1 for both Sep14 and Feb15. From the log files gathered during the experiments, we extracted several event types for all users during the first problem. The events of interest revolve around WOE usage; starting the WOE, stepping through the WOE, terminating it (quitting and completing), as well as important events including the correct and incorrect submission of solutions for problem 1. These have been cleaned and augmented with various features gathered from the individual students, such as the learning gain which was acquired via the pre/post tests. Figure 7 shows a plot of these events for the Feb15 group (described in the key in Figure 6) in a series of timelines, whereby the x-axis represents time through the problem, and each row contains events associated with an individual student. The ordering of the timelines starts from the highest gainers at the top, to the lowest at the bottom. This graph shows some common themes between users, some regardless of their achieved learning gains. Users whom utilized the WOE feature of the ITS tended to follow a pattern of starting a WOE early on in the problem, but would then quit the example early and start working on the problem again. However, they would return to the example again soon after. This may be where students may have only used the example up to the point that they felt they needed to exercise a skill, and then return to the example if they became stuck. The complete usage of examples appear to be clustered around the earlier part of the problem, with very few occurrences later on.

|	Example Started	-	Example Step
△	Example Completed	🟩	Correct Solution Submitted
✕	Example Quit	□	Incorrect Solution Submitted

Figure 6: Key for timelines

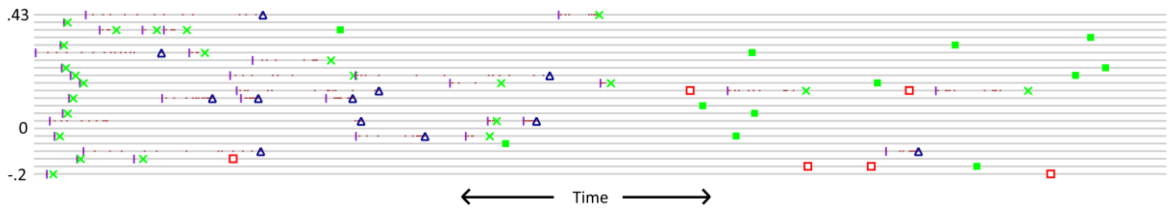


Figure 7: Event timeline for Feb15 WOE users (ordered by learning gain)

Figure 8 contrasts with that of Figure 7 by showing events from the WOE group in Sep14's experiments. There is clearly different behavior in this experiment as examples appear to be used, and completed, at all stages of the problem, and there does not appear to be a particular point that students would tend to use the example feature. The discrepancies here may help in understanding what constitutes good and bad example usage, or may identify students that may need extra structure while learning.

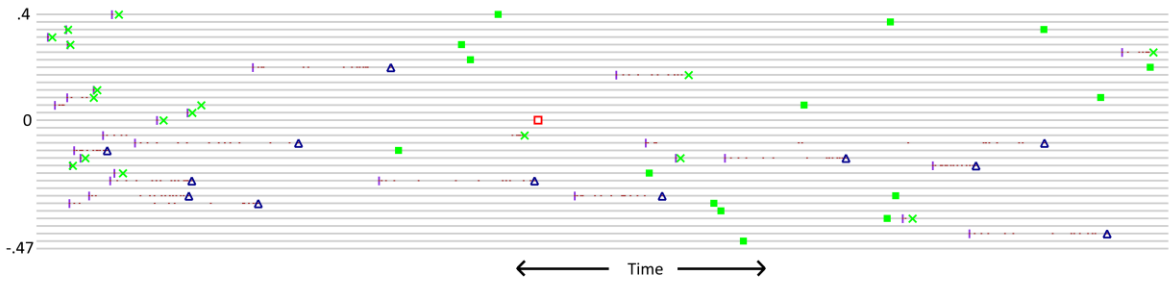


Figure 8: Event timeline for Sep14 WOE users (ordered by learning gain)

Furthermore, Figure 9 plots all students in the Sep14 conditions, once again ordered by high to low gainers. As can be seen, there is a clear pattern relating to WOE usage as all but one of the students that completed a WOE ranked in the bottom third in terms of learning gain. Although WOE students in Feb15 benefited from using WOEs in a certain manner, Sep14 students clearly did not.

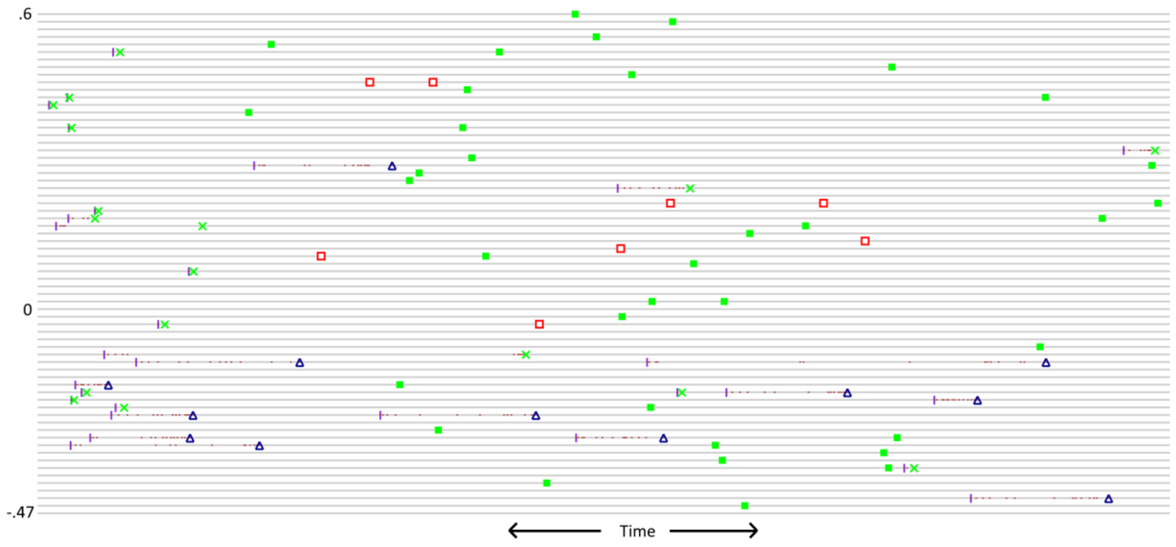


Figure 9: Event timeline for all Sep14 users (ordered by learning gain)

6. CONCLUSIONS AND FUTURE WORK

We described progress made in integrating WOEs into ChiQat-Tutor, where we implemented an engine, as well as authored relevant examples for each of the seven available problems. From here, we evaluated our work by carrying out two similar studies in September 2014 and February 2015. The experiments revealed inconsistency between the two WOE groups, where one performed very well, whilst the other made no learning gains whatsoever.

Furthermore, we looked at how students used WOEs. There were behavioral differences between the two WOE groups; the lower gaining group would tend to use examples throughout the problem, whereas the higher gaining group would appear to finish with examples early in the problem. Interestingly, the higher gaining group had twice the retention of using examples on subsequent problems. This may indicate that the high gaining group was able to learn effectively given a minimal structure while using the examples feature. Thus, the lower gaining group may benefit from additional intelligent tutoring intervention by providing examples at the optimal time. The importance of this is further emphasized by the fact that example were not just ineffective on the lower gaining group, but appeared to be even damaging with no learning gain, and virtually all students that utilized this feature ranked in the bottom third in terms of learning gain when compared with all other students in the Sep14 round of experiments.

The next step is to further understand these emerging patterns of WOE usage, and how the high and low gainers differ. We have seen some interesting patterns through visualizations, however, computational models or regression analysis may shine more light into what may be happening.

Once we understand the differences between the high and low gainers, we can then investigate possible strategies that we could put in place within our ITS that may stop inappropriate behavior and provide a more effective tutoring session for the student. This may lead to additional structure for some students, perhaps via only allowing examples to be used when deemed necessary by the intelligent tutor, or even gamifying the process (Gonzalez, Mora, & Toledo, 2014) by framing an example as a commodity.

ACKNOWLEDGEMENT

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.

REFERENCES

Atkinson, R. K., Derry, S. J., Renkl, A., & Wortham, D. (2000). Learning from examples: Instructional principles from the worked examples research. *Review of educational research*, 70 (2), 181–214.

Di Eugenio, B., Chen, L., Green, N., Fossati, D., & AlZoubi, O. (2013, July). Worked out examples in computer science tutoring. In *AIED 2013, 16th international conference on artificial intelligence in education*. Memphis, TN.

Di Eugenio, B., Fossati, D., Ohlsson, S., & Cosejo, D. (2009). Towards explaining effective tutorial dialogues. In *Cogsci 2009, the annual meeting of the cognitive science society*. Amsterdam, The Netherlands.

Fossati, D., Di Eugenio, B., Ohlsson, S., Brown, C., & Chen, L. (2015). Data driven automatic feedback generation in the iList intelligent tutoring system. *Technology, Instruction, Cognition, and Learning (TICL), Special Issue on Role of Data in Instructional Processes*, 10(1), 5–26.

Gonzalez, C., Mora, A., & Toledo, P. (2014). Gamification in intelligent tutoring systems. In *Proceedings of the Second International Conference on Technological Ecosystems for Enhancing Multiculturality* (pp. 221–225). ACM.

Green, N., AlZoubi, O., Alizadeh, M., Di Eugenio, B., Fossati, D., & Harsley, R. (2015, May). A scalable intelligent tutoring system framework for computer science education. In *CSEDU 2015, 7th international conference on computer supported education*. Lisbon, Portugal.

Lewis, C. (2010). Attrition in Introductory Computer Science at the University of California. Berkeley.

McLaren, B. M., Lim, S.-J., & Koedinger, K. R. (2008). When is assistance helpful to learning? Results in combining worked examples and intelligent tutoring. In *Intelligent Tutoring Systems* (pp. 677–680). Springer.

Miller, G. (1956). The Magical Number Seven, Plus or Minus Two. *Psychological Review*.

Najar, A. S., & Mitrovic, A. (2013). Do novices and advanced students benefit differently from worked examples and ITS? In *International Conference for Computers in Education* (pp. 20–29).

Renkl, A. (2005). The worked-out-example principle in multimedia learning. *The Cambridge handbook of multimedia learning*, 229–245.

Sweller, J. (2006). The worked example effect and human cognition. *Learning and Instruction*, 16 (2), 165–169.

Sweller, J., & Cooper, G. A. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction*, 2 (1), 59–89.