Appears in *Proceedings of SIGCSE 2003*, ACM Press, pp 6-10

Constructive and Collaborative Learning of Algorithms

Teresa Hübscher-Younger and N. Hari Narayanan
Department of Computer Science & Software Engineering
Auburn University, Auburn, AL 36849-5347
{teresa,narayan}@eng.auburn.edu

Abstract

This research began by investigating the literature on student learning from algorithm animations and conducting experimental studies of an algorithm visualization system. The results led us to develop CAROUSEL (Collaborative Algorithm Representations Of Undergraduates for Self-Enhanced Learning), using which students created expository representations of algorithms, shared their representations with others, evaluated each other's representations and discussed them. The system and the activities of representation creation, sharing, evaluation and discussion that it supports were then studied in three experiments, which are summarized. They show a significant positive relationship between these constructive and collaborative activities and algorithm learning, which suggests that this is a beneficial pedagogical approach for introductory courses on algorithms.

Categories & Subject Descriptors

K.3.1 [**Computers & Education**]: Computer Uses in Education – *Collaborative Learning*. K.3.2 [**Computers & Education**]: Computer & Information Science Education - *Computer Science Education*.

General Terms

Algorithms, Experimentation, Human Factors.

Keywords

Algorithm, Animation, Pedagogy, Multimedia, Computer-Supported Collaborative Learning

1 Introduction

Despite a research history spanning more than a decade, attempts to demonstrate the power of algorithm animations as learning tools for students have generally produced disappointing results [9]. In many cases interesting algorithm animations were designed and deployed [e.g., 12], but the literature lacks formal and systematic evaluations that provide compelling evidence for the instructional superiority of algorithm animations. Our research began with the aim of understanding how algorithm representations such as animations can be effectively used in educational settings, by first investigating how students actually learn from animations and other representations.

SIGCSE 2003, February 19-23, Reno, Nevada, USA. To appear.

The implicit model of learning most algorithm animations are based on seems to be a transmission model, where the media present relevant information in appropriate formats and students absorb the information through passive viewing and limited interaction. Our qualitative study of undergraduate computer-science students indicates that students regularly employ informal and collaborative meaning-building activities while learning algorithms [6]. These include studying in small groups, solving homework problems together, and explaining and helping each other with difficult concepts.

In order to explore how the self-paced learning model of a student sitting at a computer and interacting with an algorithm animation or visualization system engenders activities different from the natural, collaborative learning practices of students, we conducted three experiments with a system called HalVis. This system, which embeds multiple algorithm animations in an information-rich and context-providing hypermedia environment, proved to be an exception to the generally negative results regarding the efficacy of algorithm animations; it had been shown to be highly effective in helping students learn algorithms [3, 4].

We compared HalVis with a paper-based version of the system that contained exactly the same text, but substituted static images for the dynamic animations, questions for interactive exercises and page references for hyperlinks [5]. Thirty-three student volunteers were split into two groups: 19 used the computer-based version of HALVIS and 14 used the paper-based version. We found a significant difference in learning between the groups ($F(1, 31) = 7.02$, $p = .013$). The average learning score (difference between pretest and posttest scores) for the group with the computer-based version was 4.3 points (27% increase), and the average learning score for the group with the paper-based version was 7.2 points (45% increase). The group that learned from the paper-based version of HalVis significantly outperformed the group with the computer-based version. The text-based version of the system likely made students actively draw connections between the series of diagrams rather than passively observing an animation as in the computer-based system. Thus, multiple expository representations that require students to actively draw connections between them may be better for learning than a system that has students passively observe changes in one representation.

Another experiment comparing learning differences between groups using different expository representations of an algorithm suggested that interpretation problems affect learning [5]. At the end of the experiment, all 39 students involved answered two sets of questions that required procedural knowledge and mental simulation of the algorithm. They answered the first set relying on their memory alone, but to answer the second set, they were given

one of the representations, the pseudocode. Students made the same mistakes on the second set as on the first, despite being given additional information. They had misinterpreted the pseudocode originally, and in the absence of any feedback or alternative representation to compare their understanding with, they did not detect their misunderstanding even when the pseudocode was available. This indicates that misconceptions resulting from interpreting a particular expository representation may not be correctable by looking at the same representation again. Having multiple expository representations to study from may alleviate this problem.

We also conducted a usability test of HalVis, in which eight students were observed learning the Quick Sort algorithm from the system [7]. We found that all students misinterpreted at least one of the representations of the algorithm presented to them. Particularly problematic was the interpretation of the graphical representations of data structures and algorithm operations in animations. Presenting students with animations of an algorithm is supposed to help them be able to mentally visualize the algorithm. However, successfully interpreting the graphical representations they are shown can often be much more difficult than anticipated by algorithm animation designers.

The three main findings of these studies – that students find it difficult to interpret graphical representations of data structures and operations in animations created by experts; that exposure to multiple representations may help students identify their misconceptions; and that the process of actively drawing connections between multiple expository representations can benefit learning – led us to believe that having students themselves create, share, evaluate and discuss expository representations of algorithms is likely to be an effective approach to algorithm learning.

2 CAROUSEL

We designed a system called CAROUSEL to support students engaging in these four activities of constructive and collaborative learning. It is a Computer Supported Collaborative Learning (CSCL) system, which allows students to display their representations, to rate those of others and to engage in discussions about representations. It also has features for the instructor of a course to help manage course activities.

CAROUSEL is implemented using MySQL, PHP and HTML. An Apache server stores multimedia representations and a database with student and instructor information, activity information, representation store, evaluation information, comments and responses, grades for activities, etc. A web-based interface created using PHP allows instructors to set up schedules, post activity information and provide feedback to students about their performance. The interface also allows students to submit representations, evaluate representations, and comment on the representations and others' comments. CAROUSEL allowed students to share dynamic representations, provided constant access to the representations, and allowed the instructor/experimenter to implement scheduled stages of the activity – none of which would have been possible with activities using paper-based representations such as posters or storyboards.

2.1 User profiles

All users of the system have editable system profiles. Instructors and teaching assistants have more complex profiles than students. The basic system profile, the one a student has, consists of five components: user id number; name; an anonymous user name that is used to hide the student's identity; email address; and password. The instructor profile differs in that it also has information about his or her title, office location, office hours and phone number.

The system requires students to login, so that their work is kept within the class, rather than open to everyone. Logging in and tracking the user also allow the system to know which student is trying to check his/her performance or grade, turn in a representation or evaluate different representations. This permits students to edit submissions they make until a deadline has passed. Other advantages of the login facility are to allow students to access personal class information such as grades, to prevent students from misrepresenting themselves, and to prevent them from replacing other students' work. The login system also allows the teacher to provide feedback to individual students confidentially.

2.2 Managing a course

An instructor first creates a profile on the system, and then a course must be created. To set up a course, the system allows the instructor to give each course a name, an official course title, a course number, course time and location. A class roster must be built with the system as well. This roster is then used to allow the students to create profiles for using the system.

A course schedule is also entered, which is used to create the course web site. Attached to the schedule are course activities. In the case of how we have used CAROUSEL in an algorithms course, each activity consisted of students taking tests to determine how well they understand an algorithm, creating and sharing with the entire class representations of that algorithm, and evaluating and commenting on each other's representations. Therefore, information on an activity maintained by the system includes the following: activity id; activity description; directions; due dates for its different components – taking tests, submitting representations, doing the evaluations and commenting on representations; pretest and posttest grades; and the number of points/credits to be attributed to parts of an activity. The system helps the instructor and the experimenter schedule these activities along with other, more traditional activities such as homework assignments and exams.

The system provides tools for instructors to post grades for tests, points for activities, reminders for the entire class, as well as messages for individual students. In this way, CAROUSEL gives the instructor tools to provide feedback to students about their performance on course activities.

Students are able to view their current standing in the course based on the grades posted to the system, how many points have been received for each course activity, and feedback from the instructor. For participating in experiments with the system, students received extra credit points for representation creation, evaluation and pre/post test taking. Students were also able to learn what their score was for these tests (even though their test scores did not affect the credit they received for participation) in order to give them feedback about how well they understood the algorithms for which they created, shared and evaluated representations.

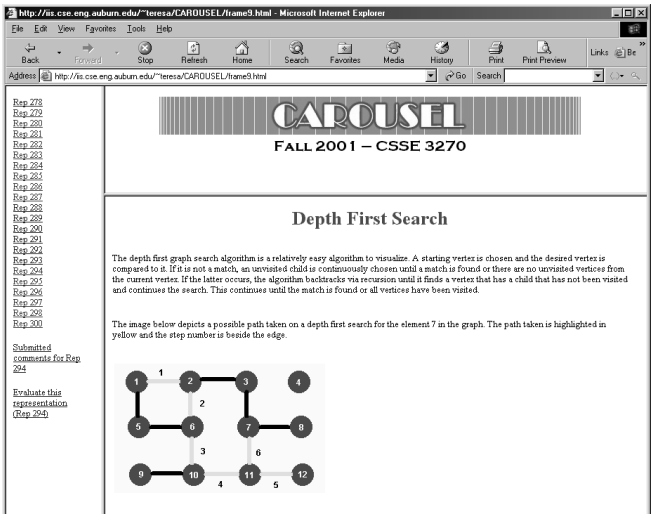
The system also compiles and displays information about how the class has been performing. Scores, averages, maximum scores and minimum scores are available. Students can see this information to see how well they are performing compared to their peers, but the information about who received which score is kept anonymous. This information is presented to students through tables, using color to highlight maximum, minimum and average scores.

2.3 Representation creation, sharing and evaluation

CAROUSEL was created, so that students could create, share and discuss multiple representations of different algorithms. An activity is set up, so that a student has a set amount of time to create a representation and turn it in. Then, the class can view the submitted representations, evaluate, and discuss them using the system over a set time period. Each activity covered one algorithm.

A student submits a representation by uploading a directory containing all the needed files to a specified directory on the Apache server. Then the student submits her directory name to CAROUSEL, which copies the files in the directory to a secure location. The student can make multiple submissions, with the later submission replacing an earlier one, until the activity due date.

Figure 1: Three frames are used to allow students to access the representations and the evaluation forms and comments attached to them.



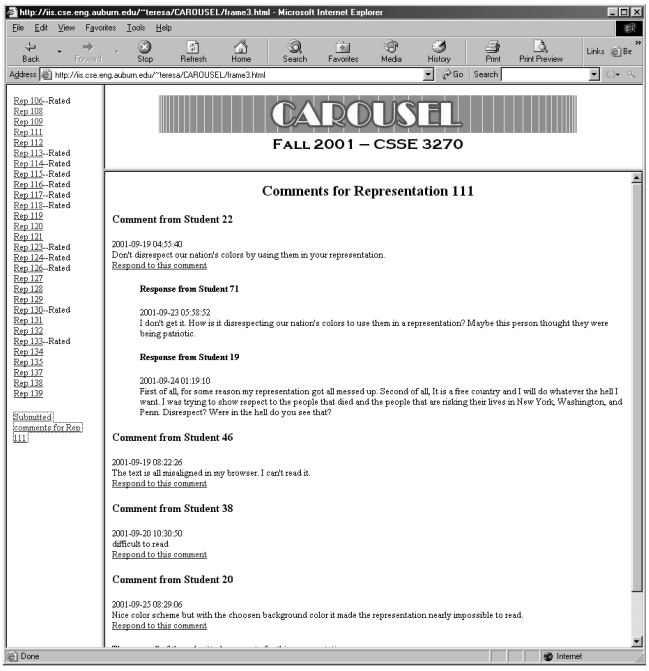
Once a student uploads a representation for an activity or assignment, the system makes it available to all other students and labels it with a number. For the first CAROUSEL experiment, representations remained numbered, until after the deadline for doing evaluations had passed. Then, the system revealed the name of the authors. For the next two experiments, the author names were not revealed. (The system allows the teacher to decide whether the representations' authors should remain anonymous, or whether and when it should be revealed).

The representations are displayed in a page with three frames for each algorithm or activity. The top frame provides system and course names. The left frame holds the list of links to the representations that are assigned by number. When one of the links is activated, the (largest) right frame is filled with the representation, and two links on the left are updated – one to get to the evaluation form for that particular representation and another to get to the submitted comments and/or current evaluation numbers for that representation (Figure 1).

Once representations for an assignment are submitted, the system allows students to evaluate these representation based on six characteristics using a five-point Likert scale. These characteristics are:

1. *Usefulness* (How central was this representation to your understanding of the algorithm?)
2. *Understandability* (How easy was this representation to understand?)
3. *Salience* (How well did this representation point out the important features of the algorithm?)
4. *Familiarity* (How familiar were you with the content of the representation?)
5. *Pleasure* (How much did you enjoy the way this representation communicated the algorithm?)
6. *Originality* (How much did this representation differ from the other representations?)

Figure 2: Students can use the system to both comment on a representation and respond to comments.



The evaluation form associated with the last viewed representation is viewed in the same frame structure as the representation. Each student is also able to comment on each representation. Each comment is labeled with the anonymous user identification number of the student who made the comment. Once a comment has been made through the evaluation form, any student or the instructor can respond to the comment. Responses can be made to responses, so a nested dialog can be attached to a comment that is attached to a representation (Figure 2).

The amount of time students have to evaluate and comment on each representation is constrained (this interval is defined by the instructor or the experimenter). Once the deadline for evaluation has passed, the link for evaluation of representations of that activity is removed from the site. The time students have to respond to comments is also similarly constrained. The system will automatically email students with reminders about deadlines for evaluation, commenting and responding to comments.

The evaluation results and comments for each representation can be made viewable to all students, according to the instructor's wishes. In the first study we conducted with CAROUSEL, color-coded statistics such as the averages, maximum score and minimum score for each characteristic for each representation were available for the students to view, along with the names of the representations' authors. However, we found that this

encouraged students to converge on a style of representation based on classroom conventions. Therefore, for our other studies with the system, we disabled this feature.

3 Three Experiments with CAROUSEL

In all of the following studies, students could choose to participate in different parts of the study for varying amounts of extra-credit. They had to take the pretests and posttests if they participated, but they could choose whether to create a representation or whether to evaluate the other representations. This was done for practical reasons, since the studies ran over a period of weeks and involved a considerable amount of work from the students.

The experiments generally had four phases:

1. The students would take a pretest and receive instructions.
2. The students would create and post their representations.
3. The students would consider and evaluate the representations posted.
4. The students would finish evaluating and commenting on each other's responses to the representations and take a posttest to see how much their understanding of the algorithm had improved.

In the first experiment there was no pretest, because it was assumed the students in a beginning data structures course had no prior knowledge of the algorithms used. Therefore we do not state the results of this study in terms of "learning" scores, since we did not measure the students' prior knowledge. The other two experiments involved students in an introductory algorithm design and analysis course, so pretests were used.

3.1 First study

This was a small study involving 12 students in a beginning data structures course. The students created and shared representations of three algorithms over a four-week period. For two of the three algorithms that were used here, there was a significant positive correlation between creating and sharing a representation and posttest scores ($r=.635$, $p=.07$; $r=.663$, $p=.05$).

This study also showed that students were converging on a representational style that was consistent with the representational style used in their lectures and textbooks [6, 8]. At the beginning of the study, the students used a variety of media in creating the representations, but by the end of the study, the only media used were text and static graphics.

CAROUSEL was subsequently revised to encourage representational and media divergence, and to lessen the effects of revealing student identities. In this study we had used *contiguity* (How well did this representation connect with the other representations for this algorithm?) as the sixth characteristic instead of *originality*. We suspected that this characteristic was unintentionally encouraging students to converge on a single representational style. So contiguity was replaced with originality. Another aspect encouraging convergence might have been the system's display of average ratings each representation received on each characteristic, compelling students to mimic styles that received high average scores. The system was revised to hide this information, and instead show only raw scores and to allow students to post public comments and responses on the representations. Finally, representations and comments were made anonymous.

3.2 Second study

This study had 60 student volunteers taking an introductory algorithms course. They created and shared representations of nine algorithms over a 12-week period. Unlike the first study, students were given a pretest and a posttest for each algorithm, and learning was measured as the difference between these scores for each algorithm. Like the previous experiment, overall results implied that the activities of creating, sharing and evaluating algorithm representations aided learning: Students improved their score from pretest to posttest by 30% on average. There was a significant difference between the normalized posttest scores ($F(1, 327)=14.4$, $p<.001$) and the normalized learning scores ($F(1, 327)=3.63$, $p=.058$) of those who had and those who had not created representations (some students did not create representations, as participation in any part of the activities associated with each algorithm was voluntary). In both cases, students who created representations had higher scores than those who did not (i.e., those who only evaluated and commented on others' representations). A total of 196 representations were created by 36 of the 60 volunteers in the study.

The learning scores were significantly related to whether a student created a representation for two of the nine algorithms: the recursive Exponentiation algorithm ($F(1, 35)=5.06$, $p=.03$) and Quick Sort ($F(1, 34)=3.99$, $p=.05$). In the case of the Exponentiation algorithm, students who did not create a representation increased their score (pretest to posttest) by 27% on average, and the students who did create representations increased their score by 45%. In the case of the Quick Sort algorithm, the students who did not create a representation increased their score by 17%, and the students who did create one increased their score by 34%.

When multiple logistic regression analysis techniques were used to analyze how whether someone did or did not create a representation and the algorithm being studied affected normalized learning scores, it was found that the model was significant ($F(9,318)=3.37$, $p<.001$). Creating a representation had a significant positive effect on learning when the choice of algorithm was controlled for ($F(1, 318)=5.025$, $p=.026$). The average normalized learning score for students who did not create a representation was 25%, while the average normalized learning score for students who did create one was 31%.

Creating a representation seemed less likely to be associated with student motivation in this study, because it did not seem to take long for students to realize that they would get credit for creating a representation regardless of the quality of their representations. Nevertheless, the average overall rating score (the average of ratings for a representation across all six rating characteristics) is significantly positively correlated to the normalized learning score (($r=.153$, $p<.01$)($F(1, 326)=5.43$, $p=.02$)), which implies that the quality of the representation created by a student affects how much he or she learns from this activity.

In this study, we found that not only were students using different media and styles, but they were also representing different aspects of the algorithm, such as representing the pseudocode itself with graphical representations, representing efficiency through interactive comparisons with different algorithms, and showing the results of execution with a particular input.

3.3 Third study

In this study 43 students in an introductory algorithms course created representations of four algorithms over a four-week period. This study was done in a semester in which other experiments (not reported in this paper) as well as regular activities such as assignments and exams were occurring. This limited the number of weeks available, resulting in the use of only

four algorithms. The overall results are again consistent with the previous two studies, indicating that the activities of creating, sharing and evaluating algorithm representations do improve learning: Students improved their score from pretest to posttest by 40% on average. The normalized posttest scores ($F(1, 94)=5.44$, $p=.02$) and the normalized learning scores ($F(1, 94)=4.43$, $p=.04$) across all algorithms were significantly different between those students who created representations and those who only evaluated and commented on others' representations. In both cases, the students who created the representations had higher scores than those who did not. The mean for the normalized posttest scores for the students who did not create a representation was 48.2% and for the students who did create one was 62%. The mean for the normalized learning scores for the students who did not create a representation was 26.3% and for those who did create one was 40.2%.

In particular, the normalized learning scores were significantly related to whether a student created a representation for one of the four algorithms, that for generating Huffman's Codes ($F(1, 23)=5.92$, $p=.02$). The average normalized learning score for this algorithm was 57% for those students who did create representations and 28% for those who did not.

A total of 65 representations were created by 22 of the 43 volunteers in the study. When multiple logistic regression analysis techniques were used to analyze how whether or not someone created a representation and the algorithm being covered affected normalized learning scores, it was found that the model was significant ($F(4, 90)=5.71$, $p<.001$). In the model, creating a representation had a significant positive effect on learning ($F(1, 90)=7.351$, $p=.008$), and the algorithm being represented also had a significant effect ($F(3, 90)=5.899$, $p=.001$).

4 Discussion

Spurred by extant literature describing mixed results regarding the educational efficacy of algorithm animations, we investigated how students learn from such systems and the natural learning strategies of computer science undergraduates studying algorithms. What we found was consistent with theories of constructivism, social cognition and collaborative learning, which hold that understanding is mediated through society, interpretation and active individuals and communities [2, 11, 13].

Based on this, we designed CAROUSEL to support students in learning about algorithms through representation creation, sharing, evaluation and discussion. Three empirical studies with CAROUSEL were conducted. Students represented algorithms with a wide array of media and styles. Their representations included metaphoric stories involving only text, entertaining animations with sound and three-dimensional graphics, as well as more conventional text and simple graphics. Results indicated that activities of representation construction, sharing, evaluation and commenting help students better understand algorithms. In the three studies, 92%, 60% and 51% of the participants constructed at least one representation. This indicates that a broad group of students, not just the smart and motivated ones, benefit from this activity. This research also provides quantitative evidence that constructing representations significantly improves student learning of algorithms. By creating representations, students engaged in creating their own explanations and thus improved their understanding [1].

While a few others have argued that having students actively engage in representation creation can help them learn algorithms [e.g. 10], there are three aspects that set this work apart. First, it involved a staged set of activities that interleaved individual construction of explanatory multimedia representations and collaborative sharing and evaluation (which included both rating and discussion). Second, students were not only the authors but also the evaluators of representations. Third, the experiments involved detailed and in-depth analyses that provided quantitative evidence for the benefits of constructive and collaborative activities carried out as part of a curriculum on algorithms.

5 Acknowledgments

This research was supported by the National Science Foundation under contract REC-9815016.

References

[1] Chi, M. T. H., de Leeuw, N., Chiu, M.-H., & LaVancher, C. (1994). Eliciting self-explanations improves understanding. *Cognitive Science, 18 (3),* 439-477.

[2] Dewey, J. (1920). *The Child and the curriculum*. Chicago: University of Chicago Press.

[3] Hansen, S. R. (1999). *A framework for animation-embedded hypermedia visualization of algorithms*. Ph.D. Dissertation, Computer Science & Software Engineering Dept., Auburn University, Auburn, AL.

[4] Hansen, S. R., Narayanan N. H. & Hegarty, M. (2002) Designing educationally effective algorithm visualizations. *Jnl. of Visual Languages and Computing, 13 (3),* 291-317.

[5] Hübscher-Younger, T. (2002). *Understanding algorithms through shared representations*. Ph.D. Dissertation, Computer Science & Software Engineering Dept., Auburn University, Auburn, AL.

[6] Hübscher-Younger, T. & Narayanan N. H. (2002). Influence of authority on convergence in collaborative learning. *Proc. Computer Support for Collaborative Learning Conf. (CSCL 2002),* Lawrence Erlbaum, 481-489.

[7] Hübscher-Younger, T. and Narayanan, N. H. (2001a). How undergraduate students' learning strategy and culture effect algorithm animation use and interpretation. *Proc. Int'l Conf. on Advanced Learning Technologies (ICALT 2001)*, IEEE Press, 113-116.

[8] Hübscher-Younger, T., and Narayanan, N. H. (2001b). Features of shared student-created representations. Paper presented at the *Workshop on External Representations in AI-ED: Multiple Forms and Multiple Roles, 10th Int'l Conf. on Artificial Intelligence in Education (AI-ED 2001),* San Antonio, TX.

[9] Hundhausen, C. D., Douglas, S. A. & Stasko, J. T. (2002) A meta-study of algorithm visualization effectiveness. *Jnl. of Visual Languages and Computing 13(3)*, 259-290.

[10] Hundhausen, C.D. & Douglas, S.A. (2002). Low fidelity algorithm visualization. *Jnl. of Visual Languages and Computing 13(5),* 449-470.

[11] Lave, J., & Wenger, E. (1991). *Situated learning: Legitimate peripheral participation*. New York: Cambridge University Press.

[12] Röbling, G. & Freisleben, B. (2000). Experiences in using animations in introductory compute science lectures. *Proc. 31st SIGCSE Tech. Symp. on Computer Science Education,* ACM Press, 134-138.

[13] Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. Cambridge, MA: Harvard University Press. (Originally published in Russian in 1934).