Usability and Pedagogical Assessment of an Algorithm Learning Tool: A Case Study for an Introductory Programming Course for High School

Aimee Theresa Avancena and Akinori Nishihara
Tokyo Institute of Technology, Tokyo, Japan

aimeetheresa@gmail.com  aki@cradle.titech.ac.jp

Abstract

An algorithm learning tool was developed for an introductory computer science class in a specialized science and technology high school in Japan. The tool presents lessons and simple visualizations that aim to facilitate teaching and learning of fundamental algorithms. Written tests and an evaluation questionnaire were designed and implemented along with the learning tool among the participants. The tool’s effect on the learning performance of the students was examined. The differences of the two types of visualizations offered by the tool, one with more input and control options and the other with fewer options, were analyzed. Based on the evaluation questionnaire, the scales with which the tool can be assessed according to its usability and pedagogical effectiveness were identified. After using the algorithm learning tool there was an increase in the post-test scores of the students, and those who used the visualization with more input and control options had higher scores compared to those who used the one with limited options. The learning objectives used to evaluate the tool correlated with the test performance of the students. Properties comprised of learning objectives, algorithm visualization characteristics, and interface assessment are proposed to be incorporated in evaluating an algorithm learning tool for novice learners.

Keywords: K12-Computer Science Education, Fundamental Algorithms, Algorithm Visualization, Learning Tool Evaluation, Questionnaire Validation

Introduction

Research on K-12 computer science education has recently gained focus as computer science has become a regular part of the curriculum in several secondary schools and is now considered essential as any of the traditional sciences. Efforts to promote such studies are being done by groups such as the ACM (Association of Computing Machinery) and the Computer Science Teachers Association (CSTA, 2005). In a curriculum report of ACM, it was suggested that a good foundation on algorithms and their implementation is necessary in order to gain programming skills and to learn more advanced computer science concepts (ACM, 2001). And in a study that
proposed a classification of research works on computer science education, one category is on tools that assist teaching, learning, and assessing programming skills (Sheard, Simon, Hamilton, & Lonnberg, 2009).

In relation to these, the study presented in this article aims to facilitate learning of algorithms among the students of an introductory computer science course through an algorithm learning tool. The first objective is to determine if the tool has an effect on the learning performance of the students. Another objective is to determine whether a visualization tool with more input options and control has more effects on learning compared to one which offers less input and control options. Along with these goals is the need to determine the scales and items for evaluating an algorithm learning tool in terms of its usability design and pedagogical effectiveness that is appropriate for novice learners. Hence, an evaluation questionnaire for the learning tool was designed and was conducted among the participants. This study would also like to find out if there is a relationship between the design properties incorporated in the algorithm learning tool and the performance of the students in the written tests in algorithms. This is in line with the plan to build a model that incorporates design properties of an algorithm learning tool with visualization and learning performance.

The next section of this paper presents related studies, particularly, on Algorithm Visualization. The third section describes the algorithm learning tool. The fourth and fifth sections present the research design and the results and discussions, respectively. These are followed by the sections on conclusion and future and suggested research plans.

**Related Work**

The learning tool presented in this paper uses the concept of Algorithm Visualization or AV, which is a technology that incorporates graphics and animation of algorithms. The algorithm process is simulated through graphical images that can be controlled by the user (Shaffer et al., 2010). AV is considered to be a subclass of software visualization which includes the development and evaluation of methods for representing software graphically, its structure, execution, and evolution (Diehl, 2007).

There is a widespread belief that Algorithm Visualization helps improve computer science education, according to previous surveys (Naps et al, 2002). This is related to the primary objective of AV, which is to aid computer science students to understand algorithms better (Hundhausen, Douglas, & Stasko, 2002). Studies have shown that visualizations and animations of algorithms and data structures are incorporated in the curriculum of introductory computer science as these are considered fundamental topics in CS education (Shaffer et al., 2010).

**Overview of History of AV**

Depiction of algorithms using graphical illustrations began in the 1960s but it was the Sorting out Sorting video created by Ron Baecker that commenced research in AV technology (Baecker & Price, 1998; Stasko, Hundhausen, Fincher, & Petre, 2004). A number of AV tools were developed since then and were unofficially classified as either pre-Java or post-Java AVs (Shaffer, et al., 2010). Pre-Java AVs came as packages with pre-generated visualizations. Some of the more known pre-Java AVs are: (1) BALSA (Brown ALgorithm Simulator and Animator) (Brown & Sedgewick, 1984); (2) TANGO (Stasko, 1990); and (3) POLKA (Parallel program-focused Object-oriented Low Key Animation) (Stasko & Kraemer, 1993).

Post-Java AVs, on the other hand, are distributed as collections of AVs that are platform independent. The more popular ones are: (1) ANIMAL (A New Interactive Modeler for Animations in Lectures) (Rößling, Schüler, & Freisleben, 2000); (2) JAWAA (Java And Web-based Algorithm Animation) (Pierson & Rodger, 1998); (3) JHAV’E (Java-Hosted Algorithm Visualization
Environment)—another Java-based system and runs on a client-server environment (Naps, Eagan, & Norton, 2000); and (4) TRAKLA2—a project of the Software Visualization Group at Helsinki University of Technology (Korhonen, Malmi, & Silvasti, 2003).

More comprehensive reports of existing and non-accessible AVs are available (Saraiya, 2002; Saraiya, Shaffer, McCrickard, & North, 2004). The Algorithm Visualization (AlgoViz) Portal (http://algoviz.org) created by Virginia Tech University is also a good resource on AVs and has links to existing AV systems.

The aforementioned AV systems are mostly highly sophisticated and are used by university students. Other visualization projects have been tried among younger learners but instead of AV, these systems employed program visualization (PV). AV and PV are considered distinct as the former deals more with the abstract descriptions of programs or algorithms while the latter aims to improve understanding of actual program implementation (Price, Baecker, & Small, 1998).

A PV system that was designed to teach Java programming to high school students is Jeliot 2000 (Ben-Bassat Levy, Ben-Ari, & Uronen, 2003). Jeliot 3, an extension of Jeliot 2000, provides visualization of the source code and a generated animation of the program’s behavior. It is intended to visualize the basic features of the Java programming language in introductory programming courses (Moreno, Myller, Ben-Ari, & Sutinen, 2004).

Another PV system, VILLE, was created for an introductory programming course at the University of Turku in Finland. Providing features such as language independency, parallel execution of a program in two different languages, and capability to define new languages, VILLE gives learners an environment to study actual program execution. The study that implemented VILLE focused on the tool’s effectiveness on learning basic programming concepts. Results indicate that the tool was beneficial for novice learners without programming background (Rajala, Laakso, Kaila, & Salako, 2008).

A recent study which incorporates both Algorithm and Program Visualizations was conducted by Osman and Elmusharaf (2014). They developed a visualized learning environment where the visualization engine was embedded within the program script instead of an external visualizer. Their results indicated that the system was effective in enhancing the performance of college students doing a course in data structures (Osman & Elmusharaf, 2014).

Pedagogical Effectiveness of AV

Most of the research works on pre-Java AVs were focused on introducing innovative features such as displays, specification techniques and interaction techniques. However, in the mid-1990s research on AV shifted to its educational contribution (Saraiya, 2002; Saraiya et al., 2004). Since then, the impact of AV on learning has been a topic of several studies in computer science education (Grissom, McNally, & Naps, 2003).

A meta-study by the group of Hundhausen (Hundhausen et al, 2002) analyzed twenty-four (24) experimental studies on Algorithm Visualization and proposed a taxonomy of scenarios of use of AV in computer science education enumerated below. Each of these scenarios distinctively describes an educational use of AV technology.

1. Lectures. Computer science instructors both in high school and college have used AV to illustrate algorithm operations in a lecture (Brown, 1988).
2. Assignments. Teachers give students assignments that require the students to construct their own visualizations of the algorithms under study (Stasko, 1997).
3. Class discussion. The students present visualizations they constructed to their classmates and instructor for feedback and discussion.
4. **Laboratories.** The laboratory exercises which incorporate AV allow the students to explore algorithms and data structures (Naps, 1990).

5. **Study.** Students taking up computer science courses use AV to study algorithms at their own pace and at any given time. They either use their own visualizations, those constructed by others or by interactive AV software.

6. **Office hours.** This is the consultation period set by college professors and teaching assistants in which students seek help or ask questions. The instructors may use AV to detect errors in the students’ programs or use AV to answer students’ questions.

7. **Tests.** Some tests may include questions where a snapshot of the algorithm animation is used and the students need to identify the algorithm (Brown, 1988).

Experiments were also carried out to validate the effectiveness of AV as an instructional material (Grissom et al., 2003). These studies present varying results from “no significance” to positive educational impact (Shaffer, Cooper, & Edwards, 2007). Studies that showed positive impact of AV systems focus on the features that make them effective (Rößling, 2009).

A study identified four features that can improve the educational effectiveness of AV: (1) Narrative contents and textual explanations for better understanding of the graphical depictions, (2) Feedback on student’s actions to reinforce right or wrong answers for AVs that ask students to predict next steps in the algorithm, (3) Extra time to use AV for tasks that cannot be animated, (4) Advanced features such as display of algorithm behaviors, interfaces to manipulate visualizations, integrated development environments (Urquiza-Fuentes & Velázquez-Iturbide, 2009). Another study proposed some key features as characteristics of a good AV: ease of use, appropriate feedback, state changes, window management, multiple views, user control, canned examples and data input, and pseudocode (Saraiya, 2002).

Furthermore, studies have shown that a visualization that allows more control of the simulation and supports student interaction and active learning is found to be more helpful and effective (Naps et al., 2002; Saraiya, 2002; Saraiya et al., 2004). Student “engagement” then is a factor considered in order to make AV educationally effective (Naps et al., 2002) and that the manner with which the students use visualization is deemed more important than the visualizations themselves (Hundhausen et al., 2002). An “engagement” taxonomy was even identified in the working group “Improving the Educational Impact of Algorithm Visualization.” This taxonomy can serve as a framework for research works that aim to determine the pedagogical effectiveness of AV (Rößling & Naps, 2002). The said taxonomy is comprised of six categories:

1. **No viewing** - no algorithm visualization is used, just plain instruction
2. **Viewing** - several visual representations of the algorithm being studied are available for viewing
3. **Responding** - questions related to the visualization displayed by the system need to be answered by the learners
4. **Changing** - modifications of the visualization, such as setting different input values to test various cases, are allowed
5. **Constructing** - users are able to create their own visualization of the algorithm
6. **Presenting** - students are required to present visualization to an audience for feedback and discussion

The algorithm learning tool reported in this paper incorporates simple Algorithm Visualization and is specifically intended for high school students. A number of features mentioned above as
well as the unique characteristics incorporated in the design of the AV created for this study are discussed in the section on the algorithm learning tool.

**Evaluation of Algorithm Visualization**

With the role AV plays in computer science education comes the need for its proper evaluation. The features and functions that researchers have been including in AV design have to be assessed if they are pedagogically effective. Studies on AV tests and evaluations have used learning taxonomies such as (the revised) Bloom's Taxonomy and the SOLO Taxonomy in order to explain the cognitive capacities of learners and to assess whether using AV has an effect on the learning outcome (Lee & Rößling, 2010).

Together with the goal of developing pedagogically beneficial features of AV, Lee and Rößling (2010) proposed that AV can be assessed according to the three properties with which media learning instruments are usually evaluated:

1. **Symbol system** - refers to texts, graphics, sounds, and animations
2. **Interactivity** - deals with user input engagement
3. **Didactic structure** – refers to pedagogical-based system design

Among the three properties, the third needs more investigation according to Lee and Rößling (2010). They then proposed the Categories of Algorithm Learning Objective or CALO to serve as a pedagogical framework for designing and structuring AV. CALO can be used as a set of objectives for exam items and as a self-evaluating tool for learners in order to measure performance based on expected learning goals. CALO is based on the seven non-hierarchical learning objectives normally used in CS education (Lee & Rößling, 2010). These objectives include:

1. **Descriptive** - discerning and describing algorithms
2. **Demonstrative** - demonstrating algorithms with graphics or objects
3. **Decoding** - following and tracking algorithms
4. **Coding** - reproducing learned algorithms
5. **Evaluative** - analyzing, comparing, and evaluating algorithms that solve the same set of problems
6. **Appropriative** - writing a complete program; evoking, extending, or modifying learned algorithms to solve a given problem
7. **Originative** - developing own algorithms to solve unfamiliar problems (Lee & Rößling, 2010).

The proponents of CALO invite researchers to adapt these categories in their own studies. They suggest that AV researchers may consider what features and functions can be incorporated into AV to support learners’ skills based on those categories.

Another study that looked into the assessment of algorithm visualization is that of Urquiza-Fuentes and Velázquez-Iturbide (2009), which focused on evaluations that yielded positive results. They classified program and algorithm visualizations systems according to two general criteria in order to facilitate evaluation, abstraction level and implementation level. The aforementioned study considered the evaluation of systems that belong to the first criterion (abstraction level). The evaluations it conducted were mainly on interactive software systems so usability was considered using the following methodologies:
1. **Informal evaluations.** The students are normally asked about their opinion after using a system.

2. **Heuristic evaluations.** Experts are the ones who usually perform these evaluations.

3. **Query techniques.** Questionnaires are used when asking students about their opinion on different aspects of the system. Evaluators have partial control of the environment and the tasks performed by students.

4. **Observational studies.** Information is gathered observing how students use the system.

5. **Controlled experiments.** The evaluators have considerable control of the environment, the students and the tasks they perform. This kind of evaluation also provides information about effectiveness, ease of use, efficiency, and other interesting issues on top of the students’ opinions. (Urquiza-Fuentes & Velázquez-Iturbide, 2009)

As usability is considered as an important aspect of interactive systems such as Algorithm Visualization tools, this research used query technique and controlled experiment in the evaluation of the algorithm learning tool. Moreover, the Categories of Algorithm Learning Objectives were used in formulating items for the learning tool’s evaluation questionnaire and the written test on algorithms. The development of the evaluation questionnaire and the written tests will be further discussed in the Research Design section.

### Algorithm Learning Tool

The main material developed for this research is an algorithm learning tool which incorporates Algorithm Visualization. The features of the algorithm learning tool were specifically designed to suit high school students in an introductory computer science class. The tool has two main parts: lecture notes and visualizations. Unlike other visualizations that target university level CS, the learning tool presented in this paper focuses on four basic algorithms: Linear Search, Binary Search, Bubble Sort, and Selection Sort. These algorithms, except for Bubble Sort, are considered essential in introductory computer science courses (ACM, 2013). However, those four were chosen because they are included in the curriculum of the high school where the tool was implemented. Moreover, the original goal for creating the learning tool is to have an instructional and review material for the student participants who are in a special computer science course track for high school. The other algorithms included in the school’s curriculum may be incorporated in future improvements or extension of the learning tool.

The learning tool provides lecture notes with descriptions, pseudocode, and illustrations of the algorithms that are designed for novice learners. The lecture notes, which has texts both in English and Japanese, include descriptions, pseudocode, and simple illustrations of the algorithms. Part of the screenshot of the lecture notes on the Linear Search algorithm is shown in Figure 1.

For the visualization of the algorithms, some principles of Algorithm Visualization were applied. A list of the tool’s features include: “textual contents, appropriate feedback, extra time to use AV for non-animated tasks, input and control menus for the animation, display of algorithm behaviors, variable state changes, ease of use, user control, canned examples and data input, and pseudocode display”, all of which were mentioned in previous studies as characteristics of a good AV (Saraiya, 2002; Urquiza-Fuentes & Velázquez-Iturbide, 2009).
In order to examine if certain levels of engagement would affect the learning performance of the students, two types of visualizations are offered by the learning tool: AlgoVis1 — allows more input options and control and AlgoVis2 — has limited input options and control. However, both AlgoVis1 and AlgoVis2 share most of the major features of the visualization. These features are enumerated below.

a. **Input and Control Panel.** This is the menu and control area where the user can set the options for the algorithm visualization and run or stop the code simulation. AlgoVis1 allows users to select the algorithm to visualize, set the size of the array used in the animation, assign array values, and choose the speed and manner of simulation. The user can set the speed from 1 to 10 (slowest to fastest) where 1 is approximately 1 second and 10 is 1/10 of a second per step. Algorithm animation can run step-by-step by marking the checkbox “Step-by-step” or by choosing the “Step-by-step” option from the list. The size of the array used in the simulation can be from 2 to 15 elements. The values of each element may be entered by the user or be randomly assigned by the computer. When all the options have been set, the “Create Array” button can be clicked to instantiate the array. The search key needed for Linear and Binary Search algorithms is entered into the box of the same name. Corresponding buttons are provided for running and terminating the algorithm visualization. A next button is used for the “Step-by-step” execution of the algorithm. Users of AlgoVis2 can only select the algorithm to simulate and choose from five data sets with pre-assigned values. “Step-by-step” algorithm simulation is not allowed in AlgoVis2.

The aforementioned features employ the “viewing” and “changing” levels of the taxonomy of learner engagement proposed by Naps et al. (2002). Screenshots of the input and control panels of the two visualization types are shown in Figure 2.
b. **Algorithm Simulation Field.** This is the main part of the visualization where the data array used to depict search and sort algorithms are shown. This part is almost similar for both AlgoVis1 and AlgoVis2. The only difference is the height of the array elements. For AlgoVis2 the elements are of the same height while for AlgoVis1, the elements vary in height depending on the assigned numerical values.

c. **Pseudocode Display.** A C-like code of the algorithm being simulated is displayed and the code is traced during simulation by highlighting each line.

d. **Variable Display.** The changes in the local variables of the algorithm are shown here.

e. **Message Box.** A description of the line of code being executed is displayed on this box.

AlgoVis1 provides more feedback to the user compared to AlgoVis2.

The last four features of the two types of visualization are shown in Figures 3 and 4, for AlgoVis1 and AlgoVis2, respectively.
Research Design

The algorithm learning tool with algorithm visualization was implemented with the main objective of determining its effects on the learning performance of the students of introductory computer science for high school. In relation to this, suggested evaluation properties for AVs were used to assess the learning tool. Through these evaluation properties, the objective is to determine if the features of the learning tool can help improve learning performance. In particular, CALO or the Categories of Algorithm Learning Objectives was incorporated in the design of the tool’s evaluation questionnaire and the written test on algorithms which were also developed for this study. The effect of the algorithm learning tool on the student performance was measured through the written pretest and posttest on algorithms.

The scores of the students in the written tests on algorithms and their response to the evaluation questionnaire were considered in the analysis. And in order to evaluate the design of the algorithm learning tool and to assess its usability and pedagogical aspects, the questions below were formulated. The answers to these questions are presented in the next section.

1. Is there an improvement in the learning performance of students after using the algorithm learning tool?

2. Is there a difference in the learning improvement between the group that had more input and control options of the algorithm visualization and the group with limited options?

3. Are the scales and items chosen for the questionnaire appropriate for evaluating the algorithm learning tool?

4. How can the questionnaire be validated and improved to better assess the effects of the tool on the learning performance of the students?

For questions 1 and 2, the corresponding null hypotheses are that the algorithm learning tool has no effect on the learning performance of the students and that there is no difference on the learning effects of the two types of visualizations used by the two groups. Questions 3 and 4 pertain to the questionnaire specifically designed for this study. The said questionnaire was used for evalu-
ating the learning tool’s usability and pedagogical effectiveness. The subsections below give more details on the design and implementation of the learning tool and the evaluation instruments.

**Methods**

The study presented in this paper may be considered as a case study because it is focused on a single group of subjects. However, the experiment could only be conducted once due to time constraints and availability of the participants. Experimental method was used to compare the pretest and posttest scores (dependent variables) of the treatment group (Group A) and the control group (Group B) based on the AV type they used, AlgoVis1 or AlgoVis2, respectively (independent variable).

To find out if there is a general increase in the learning performance of all the students after using the learning tool, *paired-samples t-test* was conducted to compare the scores of the students in the pretest and posttest. In order to determine the differences in the learning effects between the visualization with more input options and control and the one with limited options, *independent-samples t-test* was conducted to compare the pretest, posttest and gain scores between Group A and Group B.

The students’ responses to the evaluation questionnaire on the usability and pedagogical effectiveness of algorithm learning tool were also examined. The internal consistency of the questionnaire was checked using Cronbach’s Alpha reliability test in order to determine if the questionnaire scales and items are appropriate.

The evaluation questionnaire used in the implementation of the learning tool has already undergone a revision after an initial testing of the learning tool conducted one year before. In order to continue the process of questionnaire revision and validation, exploratory and confirmatory factor analyses were performed. This method is based on the two-step model building proposed by Byrne (2001). The first step is to construct a measurement model and test it to determine if the items are good indicators of their corresponding factors or properties with which the algorithm learning tool is evaluated. The second step is to build a structural model to check the relationships between the evaluation properties and the learning performance of the students, which, in this case, is the posttest score. The steps followed are similar to the work done in developing a science motivation questionnaire by Glynn, Brickman, Armstrong, & Taasoobshirazi (2011).

**Participants**

Students from TokyoTech High School of Science and Technology were asked to participate in the study. These students belong to the Information Systems course track, which offers a special computer science curriculum. At TokyoTech High, students get to choose a specific science and technology or engineering track at the end of their first year and they pursue this until they graduate. This scheme is done in order to help them select and prepare for a major in the university.

In selecting the target participants, non-probability purposive sampling was used (Patton, 1990) because the focus of the study is on high school students who are in an introductory computer science course. The sample size is then limited to the actual number of students who belong to Class 2015 of the Information System course track. There are thirty-three (33) students in the said class. These students, who are already in their final year of high school, have already studied basic algorithms the previous year. However, it was necessary for them to review and reinforce their knowledge on fundamental algorithms as they have one more year of advanced CS topics. So a special lecture was given to them using the tool as instructional aid.

The participants were divided into two groups. The grades of the students in their previous computer science course ("Programming Technique") were used as basis for the grouping. In order to
decrease the differences between the groups, the grades were sorted from highest to lowest then a
distribution scheme depicted below was done. The numbers indicate the ranking of the students.
Seventeen (17) students were assigned to Group A, and sixteen (16) to Group B.

<table>
<thead>
<tr>
<th>Group A (Treatment group)</th>
<th>Group B (Control group)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>32</td>
<td>31</td>
</tr>
</tbody>
</table>

Group A can be considered as the treatment group and Group B as the control group. In assigning
the AV type to be used by each group, the intention was to later determine if there is a difference
in the learning performance between Group A, which used AlgoVis1, and Group B, which
used AlgoVis2. In the pre-test and pre-survey stage all 33 students were present. However, one student
from group B was absent during the self-study and posttest stage of the implementation so his
data was not considered in the data analysis.

**Evaluation Instruments**

The effects of algorithm visualization tools on learning and their usability need to be examined.
This is connected to a previous research by Urquiza-Fuentes and Velázquez-Iturbide (2009),
which considered usability as an important feature of an AV tool. The *query technique* and *controlled experiment* suggested by the said study were incorporated in the evaluation of the algo-
rithm learning tool designed for this research.

As there is no established measurement for analyzing AV tools, two evaluation instruments were
designed for this study. The first evaluation instrument is the questionnaire on the usability and
pedagogical effectiveness of the algorithm learning tool. The questionnaire originally had thirty-
five (35) items each belonging to five (5) distinct scales: General Ease of Use, Interface Assess-
ment, AV Characteristics, User’s Opinion, and Algorithm Learning Objectives.

These scales and their corresponding items were designed only for the purposes of this research.
The items for the first four scales were formulated by the authors except for the items of the last
scale which were based on the Categories of Algorithm Learning Objectives or CALO. The eight
(8) items of the last scale were patterned on the seven non-hierarchical learning objectives nor-
mally used in CS education on which CALO is based (Lee & Rößling, 2010).

The original version of the said evaluation questionnaire was conducted in a pilot implementation
of the learning tool in March 2013 among Class 2014 students of the Information Systems course
track. Based on the factor analysis results, the questionnaire was revised to include only 24 items,
dropping those items that have low factor loadings (less than 0.6) (Avancena, Nishihara, & Kon-
do, 2015). The revised evaluation questionnaire was the one used in the implementation present-
ed in this paper. Please see the Appendix.

Moreover, three factors were extracted and considered as scales for the revised questionnaire.
These scales correspond to the three properties proposed by Lee and Rößling (2010) with which
to evaluate AVs. The first scale is called “AV Characteristics”, which corresponds to the “symbol system” property. It has items that deal with general characteristics of the algorithm visualization
and its execution. The second scale is “Interface Assessment”, which includes items that deal
with the input and control menu and corresponds to the “interactivity” property. The last scale is
called the “Algorithm Learning Objectives”, which corresponds to the “didactic structure” property and includes items that are based on CALO.

The other evaluation instrument made for this research is a written test on algorithms. The contents of the test were based on the actual examination on algorithms conducted by the participating high school. The test is composed of conceptual and procedural question items on the four fundamental algorithms considered in this study. The test has three parts: identification, code completion and simulation. Four of the learning objectives from CALO were integrated in each part of the test. Part I, on identification, is composed of 10 objective items and was designed after the “Descriptive” category with items that require the student to identify the algorithms and other attributes of the algorithms. Part II uses the “Coding” category, as this part involves filling in the missing lines or parts of the algorithm. Part III addresses the “Demonstrative” and “Decoding” categories where students are asked to manually demonstrate the algorithm steps and provide the output of the algorithm as required by the said categories. Each part is equivalent to 10 points so the maximum test score is 30 points.

The written test on algorithms was conducted before and after the implementation of the learning tool as pretest and posttest, respectively. The evaluation questionnaire on the usability and pedagogical effectiveness of the algorithm learning tool, on the other hand, was conducted as a post survey among the student participants.

Procedure

The algorithm learning tool was implemented among the students in the first trimester of their final year of high school. First, the written test on algorithms which lasted for 20 minutes was conducted as pretest among the students. A week after, the students attended a 40 minute lecture on algorithms using the lecture notes part of the tool as instructional material. Two weeks later, the students had another 40 minutes to study the algorithms on their own using the algorithm visualization feature of the learning tool. Group A students used AlgoVis1 (more input and control options) and group B students used AlgoVis2 (limited options). Right after the self-study using the learning tool, the students had 15 minutes to answer the evaluation questionnaire on the usability and pedagogical effectiveness of the algorithm learning tool. The same written test on algorithms given as pretest was used as posttest.

Results and Discussions

This section presents the answers to the questions mentioned in the Research Design section. The results that correspond to questions 1 and 2 are presented in the first subsection on the students’ performance in the algorithm tests. The answers for questions 3 and 4 are stated in the second subsection on the evaluation of the algorithm learning tool.

Students’ Performance in the Algorithm Tests

The first set of results presented is the performance of the students in the written tests on algorithms. To answer the question, “Is there an improvement in the learning performance of students after using the algorithm learning tool?” the scores in the pretest and posttest were compared. Line graphs that depict the pretest and posttest scores of the students are shown in Figure 5. The dots on the blue line indicate the pretest scores and those on the red line refer to the posttest scores. The pretest scores were sorted in ascending order for visualization purposes. The vertical axis indicates total scores in the test with 30 as the highest possible score while the units of the horizontal line represent the thirty-two (32) students.
The differences in the mean scores of the students in each part of the test also indicate an increase in performance in the posttest as shown in Figure 6. The blue bars indicate the mean scores in each part of the pretest and the red bars those of the posttest.

Paired-samples t-test was used to compare the scores of the 32 students in the pretest and posttest. There was a significant difference in the scores of the students in the pretest (M=12.06, SD=7.857) and posttest (M=19.38, SD=7.732) conditions; t(31)= -9.996, and the value \(p<.001\) was derived compared to 5% level of significance. The pre- and posttest score differences in each part of the test were compared and \(p<.001\) was also obtained. This result indicates that there is a general increase in the learning performance of the students after using the learning tool. However, the implementation setup allowed all the students to use the learning tool so the null hypothesis, which indicates the possibility of the tool not having an effect on learning, cannot be fully rejected.

The result which refers to the grouping scheme is presented next. This answers the question “Is there a difference in the learning improvement between the group that had more input and con-
trol options of the algorithm visualization and the group with limited options?” Group A, the treatment group, used the visualization with more input options and control (AlgoVis1) and group B, the control group, used the visualization with limited menu options (AlgoVis2). Table 1 presents the mean scores for Groups A and B.

<table>
<thead>
<tr>
<th>Group</th>
<th>Algorithm Test</th>
<th>N</th>
<th>Mean</th>
<th>Std. Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Pretest</td>
<td>17</td>
<td>11.82</td>
<td>1.718</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>15</td>
<td>12.33</td>
<td>2.298</td>
</tr>
<tr>
<td>A</td>
<td>Posttest</td>
<td>17</td>
<td>20.06</td>
<td>1.848</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>15</td>
<td>18.60</td>
<td>2.079</td>
</tr>
</tbody>
</table>

Group A has a slightly lower mean score in the pretest but has a higher mean score in the posttest. In order to determine the differences in the increase in learning performance between the groups, the gain score was calculated by subtracting the pretest score from the posttest score. Group A, the group that used AlgoVis1, has a higher gain for the total test score and in all parts of the test compared to Group B, which used AlgoVis2. Table 2 presents the differences in the gain score average of the total test score and in each part of the test for both groups.

<table>
<thead>
<tr>
<th>Group</th>
<th>AV Type Used</th>
<th>Gain Scores (Posttest Scores - Pretest Scores)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total Score</td>
<td>Part I Identification</td>
</tr>
<tr>
<td>A</td>
<td>AlgoVis1</td>
<td>8.24</td>
</tr>
<tr>
<td>B</td>
<td>AlgoVis2</td>
<td>6.27</td>
</tr>
</tbody>
</table>

The intended differences in the input menu options and control in the two AV types were examined. The group that used AlgoVis1 has higher gain score average than the groups that used AlgoVis2. However, when independent-samples t-test was conducted, results show that the differences in the pretest, posttest, and gain scores between Group A and Group B are not significant. There may be differences in the absolute scale values between the mean scores of the two groups but these differences are very small to be statistically significant. This may indicate that there is only a slight difference in the effects of a visualization that offers more input and control options compared to the one that offers less. The null hypothesis that there may be no difference between the learning gains of the two groups cannot be fully rejected then.

The general increase in the test scores of all the students may be due to the major visualization features offered by the algorithm learning tool that are shared by both AlgoVis1 and AlgoVis2. These features, which include the algorithm simulation field, the pseudocode display, the variable and message boxes, may be considered to have effects on the students’ learning performance. Based on their posttest performance, the students have proven to be capable of performing certain tasks based on the CALO categories. The students improved in their ability to identify algorithms (“Descriptive”), to fill in missing lines of codes (“Coding”) and to provide the output of an algorithm simulation (“Demonstrative” and “Decoding”). These claims, however, cannot be fully conclusive as all students were able to use the tool. An experiment set up with a control group which will not use the tool or simply receive a traditional teaching method may be carried out in a future reimplementation of the learning tool in order to verify those claims.
The results further suggest that the algorithm learning tool’s design complied with the “viewing” category of the taxonomy proposed by Rößling & Naps (2002). However, the AV type, AlgoVis1, that allows more “changing” did not prove to have a considerable difference in effect from AlgoVis2, which offers less input options. It was noticed that the students that used AlgoVis2 seemed to have a more orderly way of learning during the individual study time using the learning tool as they were given only five data sets with which to try the four algorithms. They ran each algorithm using each data set unlike the students that used AlgoVis1 who could create their own data arrays. From this observation it can be deduced that for novice learners, such as high school students, learning may also be effective if they are provided with data sets to play with and not only when given the freedom to input their own data. This observation may also be considered and needs to be proven in another implementation of the learning tool.

**Evaluation of the Algorithm Learning Tool**

The responses of the students to the questionnaire on the usability and pedagogical effectiveness of the algorithm learning tool were analyzed based on the last two questions stated in the previous section. The answers to the questions, “Are the scales and items chosen for the questionnaire appropriate for evaluating the algorithm learning tool?” and “How can the questionnaire be validated and improved to better assess the effects of the tool on the learning performance of the students?” are presented here. These two questions correspond to the process of questionnaire revision and validation.

Using Cronbach’s Alpha reliability test on all questionnaire items, the Alpha value .904 was obtained, which indicates ‘excellent’ internal consistency (George & Mallery, 2003). The same test was run to check the reliability of each scale of the questionnaire. The “Algorithm Learning Objectives” scale has an Alpha value of .896 and “AV Characteristics” has .868, which are considered ‘good’. “Interface Assessment” has an Alpha value of .744 describing ‘acceptable’ internal consistency. Table 3 shows the reliability values of the three scales based on the responses of all the students.

<table>
<thead>
<tr>
<th>Category</th>
<th>Alpha Reliability (N=32)</th>
<th>No. of Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>AV Characteristics</td>
<td>.868</td>
<td>9</td>
</tr>
<tr>
<td>Interface Assessment</td>
<td>.744</td>
<td>7</td>
</tr>
<tr>
<td>Algorithm Learning Objectives</td>
<td>.896</td>
<td>8</td>
</tr>
</tbody>
</table>

The two-step model building proposed by Byrne (2001) was implemented next in order to determine if the questionnaire items are good indicators of their corresponding factors. Exploratory factor analysis was conducted first on the evaluation questionnaire. Maximum Likelihood was used for extracting the factors and Promax with Kaiser Normalization for rotating the factors. To see if the resulting matrix of correlations was appropriate for factor analysis, Bartlett’s test of sphericity was determined and yielded the following values: Chi-square = 563.961, df = 276, p < 0.001, and a Kaiser–Meyer–Olkin measure of sampling adequacy, KMO = 0.522.

Table 4 shows the result of the factor analysis of the revised evaluation questionnaire. Three factors were extracted which correspond to the same three factors extracted for the original version of the questionnaire. So the same names of the three factors can still be used: Algorithm Learning Objectives, AV Characteristics, and Interface Assessment. However, there are slight differences in the items that loaded into the three factors compared to the grouping of items before factor analysis and some of the factor loadings are quite low. Nonetheless, the result still suggests that the use of the CALO taxonomy was suitably incorporated in the design of the questionnaire items.
The second step in structural modeling was carried out to examine the relationships between the three factors previously extracted and the posttest scores of the students. A measurement model was constructed and tested by means of confirmatory factor analysis using Analysis of Moment Structures (AMOS) version 18.0. This method was used in order to further examine the extracted factors and their associated items. Moreover, confirmatory analysis was used in order to establish the construct validity of the questionnaire (Pett, Lackey, & Sullivan, 2003). The standardized correlation values and factor loadings presented in Figure 7 are all acceptable, (greater than .35) except for questionnaire item U8, indicating that the factors are correlated with each other and that each item is a good measure of its corresponding factor.

Table 4. Factor Analysis of the Evaluation Questionnaire

<table>
<thead>
<tr>
<th>Questionnaire Items</th>
<th>Factor Loadings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td><strong>Algorithm Learning Objectives Factor</strong></td>
<td></td>
</tr>
<tr>
<td>U18. I can describe how the algorithms work.</td>
<td>.922</td>
</tr>
<tr>
<td>U19. I can demonstrate how the algorithm works using drawing simulations</td>
<td>.870</td>
</tr>
<tr>
<td>U17. I can now identify the algorithm by just looking at the pseudocode.</td>
<td>.749</td>
</tr>
<tr>
<td>U22. I can compare and analyze algorithms that solve the same problems, e.g.</td>
<td>.731</td>
</tr>
<tr>
<td>search and sorting.</td>
<td></td>
</tr>
<tr>
<td>U23. I can easily code the algorithms using C programming language or another</td>
<td>.702</td>
</tr>
<tr>
<td>language I know.</td>
<td></td>
</tr>
<tr>
<td>U21. I can complete the missing code for all the four algorithms I learned.</td>
<td>.671</td>
</tr>
<tr>
<td>U20. I can give the output for a set of data by using algorithm simulation.</td>
<td>.556</td>
</tr>
<tr>
<td>U24. I can now develop my own algorithms to solve other problems.</td>
<td>.536</td>
</tr>
<tr>
<td>U7. The pseudocode display is helpful in understanding better the algorithm.</td>
<td>.496</td>
</tr>
<tr>
<td>U3. The Algorithm Learning Tool and the Algorithm Visualization provide enough</td>
<td>.386</td>
</tr>
<tr>
<td>user interaction.</td>
<td></td>
</tr>
<tr>
<td><strong>AV Characteristics Factor</strong></td>
<td></td>
</tr>
<tr>
<td>U13. The Algorithm Visualization allows the user to set the size of the array.</td>
<td>.403</td>
</tr>
<tr>
<td>U12. The Algorithm Visualization allows the user to choose the speed of the</td>
<td>.497</td>
</tr>
<tr>
<td>algorithm animation.</td>
<td></td>
</tr>
<tr>
<td>U15. The Algorithm Visualization allows the user to assign the elements of the</td>
<td>.221</td>
</tr>
<tr>
<td>array.</td>
<td></td>
</tr>
<tr>
<td>U16. The Algorithm Visualization gives appropriate feedback to the user</td>
<td>.390</td>
</tr>
<tr>
<td>U14. The Algorithm Visualization asks questions about the next steps in the</td>
<td>.301</td>
</tr>
<tr>
<td>algorithm simulation.</td>
<td></td>
</tr>
<tr>
<td>U9. Setting the size and values of the array is helpful in learning the algorithms</td>
<td>.219</td>
</tr>
<tr>
<td>better.</td>
<td></td>
</tr>
<tr>
<td>U11. The Algorithm Visualization allows the user to choose the algorithm to study.</td>
<td>.264</td>
</tr>
<tr>
<td>U10. The algorithm animation is helpful in understanding how the algorithm works</td>
<td>.296</td>
</tr>
<tr>
<td><strong>Interface Assessment Factor</strong></td>
<td></td>
</tr>
<tr>
<td>U5. It is better if actual coding or programming is allowed in Algorithm Learning</td>
<td>.207</td>
</tr>
<tr>
<td>Tool.</td>
<td></td>
</tr>
<tr>
<td>U2. The instructions on how to use the Algorithm Learning Tool and the Algorithm</td>
<td>.071</td>
</tr>
<tr>
<td>Visualization are clear.</td>
<td></td>
</tr>
<tr>
<td>U4. The control buttons to start, stop and restart algorithm and to run the</td>
<td>.281</td>
</tr>
<tr>
<td>algorithm step by step are useful for learning the algorithms better.</td>
<td></td>
</tr>
<tr>
<td>U6. It would be better if there is a ‘back’ button when tracing the algorithm.</td>
<td>.039</td>
</tr>
<tr>
<td>U8. The menu choices for the Algorithm Learning Tool are adequate.</td>
<td>-.087</td>
</tr>
<tr>
<td>U1. The Algorithm Learning Tool and the Algorithm Visualization are generally</td>
<td>.341</td>
</tr>
<tr>
<td>easy to use.</td>
<td></td>
</tr>
</tbody>
</table>
Model fitting was done using Unweighted Least Squares estimation because of the small sample size. It yielded the following fit index values: GFI (Goodness-of-Fit Index) = .908; AGFI (Adjusted Goodness-of-Fit Index) = .889; RMR (Root Mean Square Residual) = .122; NFI (Normed Fit Index) = .876; and RFI (Relative Fit Index) = .862. All the given indexes indicate a good fit if the value is close to 1 except for RMR which is better if it is closer to zero. The above results indicate that the measurement model represents well the three factors that correspond to the three properties with which the algorithm visualization tools can be evaluated and provides evidence for the questionnaire’s construct validity.

A structural model was built next in order to examine the relationship between the evaluation factors and the learning performance of the students. The model is shown in Figure 8. In building this structural model, scores in the posttest on algorithms were considered. This model may be used to suggest that among the three evaluation factors, Learning Objective has the biggest effect on the posttest score (.65). Another assumption that can be made is that AV Characteristics affects Interface Assessment (.51) and Learning Objectives (.48). Interface Assessment is slightly related to Learning Objectives (.08). Using Scale-Free Least Squares estimation the following index fit values were obtained GFI (Goodness-of-Fit Index) = .876; AGFI (Adjusted Goodness-of-Fit Index) = .852; NFI (Normed Fit Index) = .833; PRATIO = .907, where an index value close to 1 indicates a good fit.
The results of the two-step modeling for the questionnaire revision and validation suggest that the three scales of the evaluation questionnaire have incorporated the proposed three properties for AV evaluation (Lee & Rößling, 2010) and that the questionnaire items are good measures of its corresponding scale. Having incorporated CALO in the design of the questionnaire, the structural model may be regarded as a response to the suggestions of Lee and Rößling (2010). As CALO has been proposed to be a didactic structure for AV designers, the model may be considered as a pedagogical framework in incorporating functions into learning tools that use AV. Another proposal for CALO is that it can be used as a set of learning objectives in writing test items that assess the pedagogical effectiveness of AV tools. The model also suggests that the posttest used in this study have incorporated the recommended algorithm learning objectives. However, the specific test items may need additional examination to find out their relationships with CALO. The model may also be further revised and validated with a larger sample size. In summary, the structural model may be used as a guide in designing an instrument to assess the usability and pedagogical effectiveness of an algorithm learning tool for high school computer science. The obtained evaluation factors may also be considered in further enhancing the algorithm learning tool developed for this study.

![Figure 8. Relationship between Evaluation Factors and Post Test Performance](image)

**Conclusions**

An online algorithm learning tool that uses Algorithm Visualization (AV) technology was designed and developed for the students in an introductory computer science course at TokyoTech High School of Science and Technology. After the implementation of the learning tool, there is an increase in the performance of the students in the written test on algorithms. This indicates that the students have proven to be capable of performing certain tasks based on the Categories of Algorithm Learning Objectives. In the absolute scale, the students that used that tool that provide more input options and control have higher increase in the posttest scores. However, the differences in the scores between the control group and the treatment group are not big enough to produce statistically significant results.
The questionnaire designed to evaluate the algorithm learning tool was examined. The three factors obtained using exploratory and confirmatory factor analysis, (1) Algorithm Learning Objectives, (2) AV Characteristics, and (3) Interface Assessment, correspond to the three properties recommended for analyzing and evaluating Algorithm Visualization tools. The structural model that relates the three factors and the posttest scores of the students may be used in constructing an instrument to assess an algorithm learning tool for novice learners. The model may also be used as reference in designing such tools.

**Future Work**

As previously mentioned, another implementation of the algorithm learning together with the evaluation instruments designed for this study may be done to further prove the effects of the tool on learning and to validate and improve the questionnaire and the written tests. As there was an unavoidable limit to the sample size of participants, there is a plan to implement the tool among the next batch of students from the Information Systems course track (Class 2016). The grouping scheme may be modified such that there is a control group that will not use the tool. Another future direction involves extending the algorithm learning tool to include other fundamental algorithms.

**References**


**Appendix**

**Evaluation questionnaire on the usability and pedagogical effectiveness of the algorithm learning tool and the algorithm visualization**

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The Algorithm Learning Tool and the Algorithm Visualization are generally easy to use.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. The instructions on how to use the Algorithm Learning Tool and the Algorithm Visualization are clear.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The Algorithm Learning Tool and the Algorithm Visualization are generally easy to use.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. The instructions on how to use the Algorithm Learning Tool and the Algorithm Visualization are clear.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Interface Assessment**
3. The Algorithm Learning Tool and the Algorithm Visualization provide enough user interaction.
アルゴリズム学習ツールとアルゴリズム可視化は十分なユーザ・インタラクションを提供している

4. The control buttons to start, stop and restart the algorithm visualization and to run the algorithm step by step are useful for learning the algorithms better.
再生や一時停止、再開、アルゴリズム可視化を段階的に実行するコントロールボタンはアルゴリズムの学習に役立つ

5. It is better if actual coding or programming is allowed in Algorithm Learning Tool.
実際にプログラミングをする際にアルゴリズム学習ツールを使えばより良い

6. It would be better if there is a ‘back’ button when tracing the algorithm.
アルゴリズムの処理を追う際にバックボタンがあればより良い

7. The pseudocode display is helpful in understanding better the algorithm.
擬似コードの表示はアルゴリズムの理解に役立つ

### Algorithm Visualization’s characteristics

8. The menu choices for the Algorithm Learning Tool are adequate.
アルゴリズム学習ツールのメニュー選択は適切である

9. Setting the size and values of the array is helpful in learning the algorithms better.
配列のサイズと値をセットすることは、より良くアルゴリズムを学ぶことに役立つ

10. The algorithm animation is helpful in understanding how the algorithm works.
アルゴリズム・アニメーションはアルゴリズムがどのように機能するのかを理解するのに役立つ

11. The Algorithm Visualization allows the user to choose the algorithm to study.
アルゴリズム可視化ではユーザが勉強したいアルゴリズムを選択することができる

12. The Algorithm Visualization allows the user to choose the speed of the algorithm animation.
アルゴリズム可視化ではユーザがアルゴリズム・アニメーションの速度を選ぶことができる

13. The Algorithm Visualization allows the user to set the size of the array.
アルゴリズム可視化ではユーザが配列のサイズを設定できる

14. The Algorithm Visualization asks questions about the next steps in the algorithm simulation.
アルゴリズム可視化はアルゴリズムのシミュレーション次のステップについての質問を行う。

15. The Algorithm Visualization allows the user to assign the elements of the array.
アルゴリズム可視化ではユーザが配列の要素を割り当てることができる

16. The Algorithm Visualization gives appropriate feedback to the user.
アルゴリズム可視化はユーザに適切なフィードバックを行う

### Algorithm Learning Objectives

17. I can now identify the algorithm by just looking at the pseudocode.
私は擬似コードを見るだけでアルゴリズムを識別できる
18. I can describe how the algorithms work.
    私はアルゴリズムがどのように機能するかを述べることができる

19. I can demonstrate how the algorithm works using drawing simulations.
    私はアルゴリズムがどのように機能するかを、状況を描くことによって説明することができる

20. I can give the output for a set of data by using algorithm simulation.
    私はアルゴリズム・シミュレーションを使うことによりデータセットの出力を与えることができる

21. I can complete the missing code for all the four algorithms I learned.
    私は学習した4つのアルゴリズム全てについてコードを完成させることができる

22. I can compare and analyze algorithms that solve the same problems, e.g. search and sorting.
    私は類似する問題（すなわちサーチとソート）を解決するアルゴリズムを比較、分析することができる

23. I can easily code the algorithms using C programming language or another language I know.
    私はC言語や他のプログラミング言語を用いてアルゴリズムを簡単にコード化できる

24. I can now develop my own algorithms to solve other problems.
    私は他の問題を解決するために自身でアルゴリズムを考えることができる

---

### Biographies

**Aimee Theresa Avancena** is a PhD student at Tokyo Institute of Technology where she finished Master in Engineering in 2012. She obtained a B Computer Science degree from the University of the Philippines in 1996 and MS in Computer Science from the Ateneo de Manila University in 2008. She was a computer science teacher at Philippine Science High School from 1997 to 2009. Her interests include computer science education, educational technology and English education.

**Akinori Nishihara** is a Professor of the Department of Human System Science, Graduate School of Decision Science and Technology, Tokyo Institute of Technology. His research interests include signal processing and educational technology (especially e-learning and distance education). He has published more than 300 technical papers in refereed international journals and conferences. He is a Fellow of IEEE and IEICE, and a member of AACE, APSIPA, JSET, and JSISE.