Supporting Materials for Active e-Learning in Computational Models

Mohamed Hamada

Languages Processing Lab
The University of Aizu, Aizuwakamatsu, Fukushima, Japan
hamada@u-aizu.ac.jp

Abstract. In traditional lecture-driven learning, material to be learned is often transmitted to students by teachers. That is, learning is passive. In active learning, students are much more actively engaged in their own learning while educators take a more guiding role. This approach is thought to promote processing of skills and knowledge to a much deeper level than passive learning. In this paper, a research using supporting materials for active e-learning in computational models and related fields is presented. The contributions of this paper are supporting active tools to improve learning and an evaluation of its use in context.

1 Introduction

Active and collaborative learning provide a powerful mechanism to enhance depth of learning, increase material retention, and get students involved with the material instead of passively listening to a lecture. Active learning is a learning with students involved in the learning process as active partners: meaning they are "doing", "observing" and "communicating" instead of just "listening" as in the traditional (lecture-driven) learning style.

Learning science research indicates that engineering students tend to have active and sensing learning preferences, and engineering related educators are recognizing the need for more active and collaborative learning pedagogy [22]. So far several learning models have been developed (e.g. [4, 9, 13, 17]) for the realization of the learning preferences of science and engineering learners. Among these models, Felder-Silverman [4] is simpler and easier to implement through a web-based quiz system, as in Felder-Soloman [21]. The model classifies engineering learners into four axes: active vs. reflective, sensing vs. intuitive, visual vs. verbal, and sequential vs. global. Active learners gain information through a learning-by-doing style, while reflective learners gain information by thinking about it. Sensing learners tend to learn facts through their senses, while intuitive learners prefer discovering possibilities and relationships. Visual learners prefer images, diagrams, tables, movies, and demos, while verbal learners prefer written and spoken words. Sequential learners gain understanding from details and logical sequential steps, while global learners tend to learn a whole concept in large jumps.

In Rosati [20] a study of this model was carried out to classify the learning style axes of engineering learners. The study showed that engineering learners tend to have strong active, sensing, visual, and sequential learning preferences.

M. Bubak et al. (Eds.): ICCS 2008, Part II, LNCS 5102, pp. 678 – 686, 2008.

The concepts of computational models have important use in designing and analyzing several hardware and software applications. These concepts are abstract in nature and hence used to be taught by a traditional lecture-driven style, which is suitable for learners with reflective preferences. Since computer engineering learners tend to have strong active preferences, a lecture-driven teaching style is less motivating for them.

In this paper, a research using supporting materials for active e-learning in computational models and related fields is presented. The contributions of this paper are supporting active tools to improve learning and an evaluation of its use in context. As a first contribution, we introduce an integrated environment that is designed to meet the active learning preferences of computer engineering learners. For the second contribution: several classroom experiments are carried out. The analysis of the experiments' outcomes and the students feed back show that our integrated environment is useful as a learning tool, in addition to enhancing learners' motivation to seek more knowledge and information on their own.

Our active materials can be used as a supporting tool for active (e)-learning not only for computational models subject, but also for several other courses such as automata and formal languages, theory of computation, discrete mathematics, principles of programming languages, compiler design and other related courses. Such courses cover a variety of topics including finite state machines (automata), push-down automata, and Turing machines, in addition to grammars and languages. We cover such topics in our active tools. The tools are written using the Java2D technology of Sun Microsystems [10]. This implies that our tools are portable, machine independent and web-based enabled, which makes it a useful tool as an interactive and online learning environment.

The tools integrate several different materials to support the learners' preferred style. It includes a movie-like welcome component, an animated hyper-text introduction for the basic concepts, a finite state machine simulator with several operations, a set of visual examples for learners' motivation, a Turing machine simulator, and an interactive set of exercises for self assessment.

To show the effectiveness of our tools as a model of interactive online collaborative learning tool, several classroom experiments were carried out. The preliminary results of these experiments showed that using our environment not only improved the learners' performance but also improved their motivation to actively participate in the learning process of the related subjects and seek more knowledge on their own.

The paper is organized as follows. Following the introduction, section two introduces our active tools including finite state machines, visual examples, and Turing machines. The performance evaluation of the environment will be presented in section three. Finally, we conclude the paper and discuss results, related work, and possible future extensions in section four.

2 Active Materials

Our active materials contain eight components which have been integrated into a single unit to make all topics easily accessible for learners. The components include the following: a movie-like welcome component, a hyper text introduction to the computational models topics, a finite state machine (FSM) simulator. a Turing machine

(TM) simulator, a self assessment exercises, and the other three components showing the visual examples of finite state machines. The welcome and introduction components use plain and animated text, which are suitable for learners with sequential learning preferences. The simulators and visual examples of components are best suited for learners with active and sensing learning preferences which most computer engineering learners prefer. In the sequel of this section, we will describe of these components.

2.1 FSM Simulator

The finite state machine simulator is integrated as a basic component of the environment. It allows learners to draw an automaton visually and apply several operations to it. The possible operations include: NFA to DFA transformation, λ-NFA to NFA transformation, DFA to regular expression, and regular expression to λ-NFA. In addition to these transformations, learners can minimize the given automaton, check the acceptance/rejection of an input to the automaton, zoom-in and out, and auto layout the automaton. The simulator interface is shown in Fig. 1.

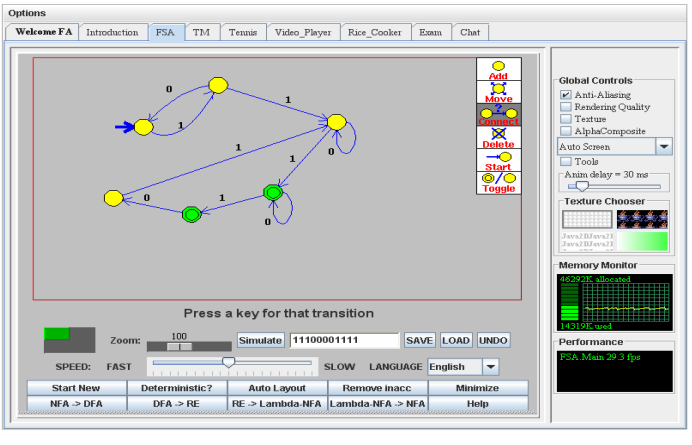
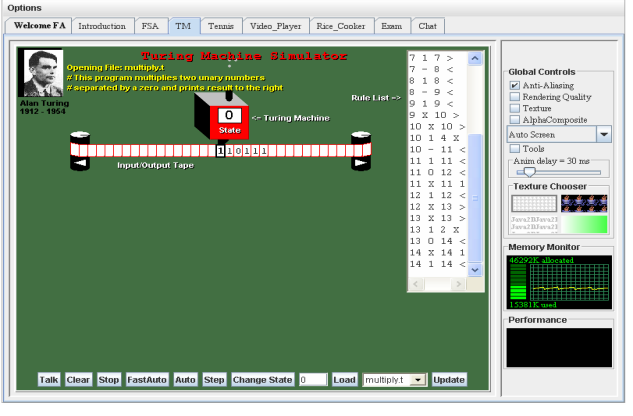


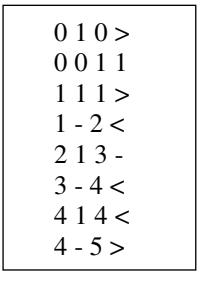
Fig. 1. The FSM simulator interface

2.2 TM Simulator

The Turing machine simulator is integrated into the environment as well. This simulator is based on the work of [11]. Learners can write their machine in the input window, and then write the input of the machine on the (infinite) tape. After that, they can start to operate the machine on the input and observe how it works. For example, to add two positive integers m and n, the function $add(m, n) = m+n$, is represented by the Turing machine rules shown in Fig. 2(b). A rule in the form $a\ b\ c >$ means that if the current state is a and the current input tape symbol is b, then the controller changes the current state to c and moves one step to the right (right is represented by > and left by <). A rule in the form $a\ b\ c\ d$ means that if the current state is a and the current

input tape symbol is *b*, then the controller changes the current state to *c* and the current input tape symbol to *d*. If the learner wants to add, for example, 2 and 3, i.e. compute the function *add*(2,3)=2+3, then he/she must write 2 as 11 and 3 as 111 separated by 0 on the input tape, which means the input string will be 110111. Running the machine on this input by clicking the run button will result in the machine halting with the output string 11111 written on the tape, which means 5. Learners can see the machine running on the input symbol in a step-by-step manner which can help the learner to see how the Turing machine acts on input and how it can compute functions. All operations of the Turing machines can be simulated by this simulator. The component interface showing the Turing machine simulator is shown in Fig. 2(a). While the Turing machine operates on its input, a number of short comments also appear on the editor to give the learners more information about the theory of Turing machines. The Turing machine simulator also includes sound effects to make learning more fun and interesting to learners.





(a) (b)

Fig. 2. (a) The TM simulator interface (b) TM example

2.3 Visual Examples

Our tools contain a set of visual examples that we introduced with the aim of motivating learners in courses that include such topics. These selected examples represent useful daily life machines, games, and a puzzle. We have created six examples: an elevator, a vending machine, a man, a wolf and a goat puzzle, a video player, a rice cooker, and a tennis game. In this section, we will describe the last one as an example.

2.3.1 Tennis Game Simulator

A poplar game such as Tennis can be modeled by automata. The following example shows such model. The input set represents the two players *a* and *b*. In this DFA there are 20 states. The start state is located at the root and denoted by in-arrowed circle.

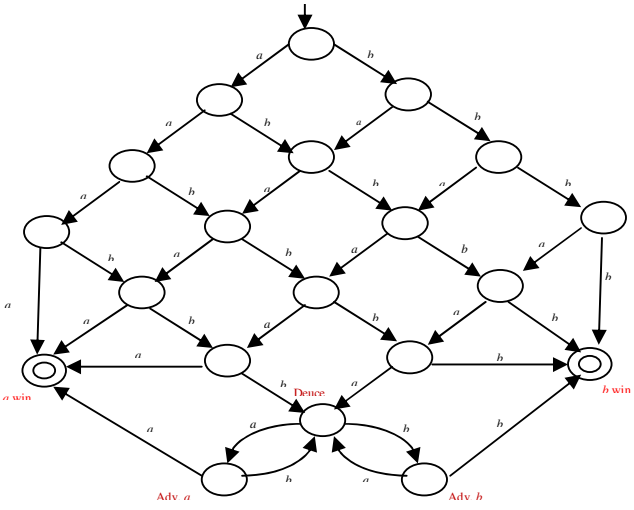


Fig. 3. An automaton representing tennis game

There are two final states denoted by double circles. The automaton enters a final state when one of the players wins the game. The transition diagram is shown in Fig. 3.

A typical tennis game can be represented by three finite automata: one for the points, one for the games, and one for the sets. Our tennis game simulator considers two players, A and B, who can be selected to play. It also allows auto play where the players play randomly. The simulator displays the score as well as the underlying three automata. The first automaton is related directly to the points; when a player wins a point, a new state is created. The second automaton is related to the game; when a player wins a game, a new state is created. The third automaton is related to sets; when a player wins a set, a new state is created. Fig. 4 shows a snapshot of the tennis game simulator interface. The game is completed when a final state of the

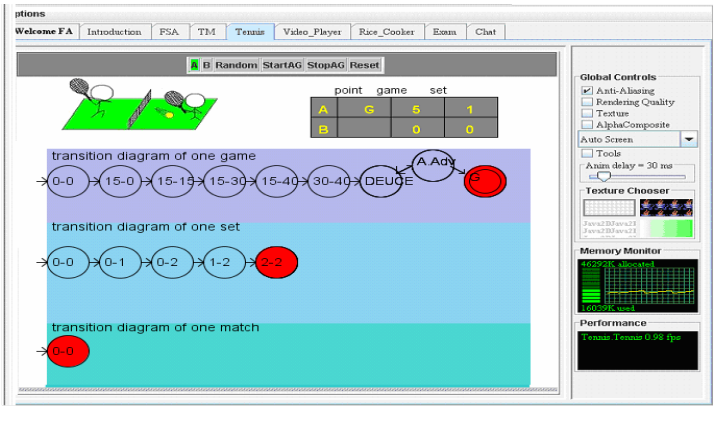


Fig. 4. The Tennis game simulator interface

automaton, representing the game set, is created. The winner is the player who reaches this final state first. The simulator also integrates animation of the two virtual players and various sound effects to make the learning more fun and interesting.

In the video player and rice cooker examples, the underlying automata are given first, and then when an operation takes place, the corresponding state is highlighted. In the tennis game automata, however, we considered a different approach: the automata are created state by state in response to the corresponding operation. This is done intentionally to give the students a different taste of machine modeling by automata and to make it more interesting.

2.4 Learners Self-assessment

A set of exercises with different levels is also integrated with the environment. There are various types of quizzes: some are multiple choice, some are fill in the blanks, and some test for Turing machines, finite automata or regular expressions. Learners can perform a pre-assessment, an in-assessment, or a post-assessment. First, the learner must select an exercise and then a description of the test and the evaluation method will be shown in the main window. Learners can navigate among the quizzes by using the navigation buttons at the bottom of the main window. Learners can check the score at any time by clicking on the 'score' button. While answering a quiz, learners can get hints or click on the introduction button on the top of the window to go to the introduction component and read more about the topics related to the quiz.

3 Tools Assessment

We carried out two experiments in order to evaluate the effectiveness of our integrated environment tools on the learning process of engineering students. The first experiment evaluates the improvement in the students' motivation. The second experiment evaluates the effectiveness of using the tools on the students' performance.

The purpose of introducing the visual automata examples is to enhance the students' motivation. To measure the effectiveness of these visual examples, we performed two experiments in the automata and formal languages course. The first one was for students who already completed the course; the sample population included 52 students who studied the topics in different classrooms. The following question was asked: "If the course was an elective course, would you choose to study it? And, do you recommend other students to study it?" Five options were given for responses: (a) don't know, (b) no, (c) maybe no, (d) maybe yes, and (e) yes. The responses were as follows: 3 answered a, 3 answered b, 6 answered c, 27 answered d, and 13 answered e. Then, we demonstrated our visual examples to the students and repeated the same question again. Their responses (after seeing the examples) were: 1 for a, 3 for b, 2 for c, 29 for d and 17 for e. Comparing the results from "Before" and "After" exposure to the examples, there was a slight improvement in motivation. For choices a, b, and c, if the number of responses decreased, it indicates a positive response, which is what occurred. While for the other choices d and e, the increasing number of responses indicates positive response, which also occurred.

We note that there was only a small improvement in the students' motivation, which is natural in this case because the students had already completed the course. In the next experiment we noted a better improvement in the motivation of students who were new to the course.

In the second experiment, a total of 69 students were included, and they were all new to the course. The same steps, as with the pervious experiment, were repeated with a slight modification in the question. The question was "If the course was an elective one would you chose to study it?" As before, students were allowed to choose from among the five responses: a, b, c, d, and e. Their responses (before seeing the examples) were as follows: 22 answered a, 6 answered b, 10 answered c, 23 answered d, and 8 answered e. Next, we demonstrated our visual examples to the students and presented the same question to them again. Their responses (after seeing the examples) were as follows: 9 answered a, 4 answered b, 8 answered c, 34 answered d, and 14 answered e. Comparing the results "Before" and "After" exposure to the examples, we can see a better improvement in their motivation. As with the previous experiment, for choices a, b, and c, if the number of responses decreased it meant a positive response, which is what occurred. While for the other choices d and e, an increasing number of responses meant a positive response, which also occurred.

We note that the motivation in the case of junior students (second experiment) was better than that of the senior students (first experiment). This result might be explained by the fact that the juniors had not studied the course before.

A preliminary study shows that the integrated environment can improve the learning process of computer engineering students who study automata theory course and related courses. Last semester, the students were divided into four groups, each group containing 20 students. A set of 40 randomly selected exercises was distributed among the groups, 10 for each group. Each group members could collaborate inside their group but not with any other group members. No group could see the exercises of other group. Two groups were asked to answer their assigned exercises using the integrated environment and the other two groups without using it. An equal time period was provided to all the groups. The result showed a better performance for the two groups using the IE. Then, the experiment was repeated by redistributing the exercises among the four groups. Again, the two groups with the IE showed better performance.

4 Conclusion

Applications of technology can provide course content with multimedia systems, active learning opportunities and instructional technology to facilitate education in the area of computer engineering to a broad range of learners. Such interactive course materials have already been introduced for several topics in computer engineering courses; see for example [5, 6, 7, 14, 15, 18].

In this paper, we followed the same path and introduced a set of visual tools to support interactive (e)-learning for computational models concepts. It can also be used in other courses such as automata and formal languages, language processing, theory of computation, compiler design, discrete mathematics, and other similar courses.

There are a number of similar tools which have been developed (e.g. [1,2,3,8, 16, 19]) to enhance the learning of computational models topics. Most of them suffer from one or more flaws that make them less effective as a learning tool, particularly for less advanced students. For example, JFLAP [19] is a comprehensive automata tool but it requires skilled learners who already know the basics of automata to make full use of its rich operations. The automata tools in [16] are a powerful tool, but do not provide a convenient mechanism for displaying and visually simulating the finite state machines. The ASSIST automata tools in [8] are difficult to setup and use. The tools in [1] lack visual clarity and dynamic capability. Almost all have been designed as tools for advanced learners. These tools work on the assumption that the learners have already grasped the fundamental concepts. They are also dependent on advanced mathematical and idiosyncratic user interactions. On the contrary, our tools are designed as an easy-to-use, easy-to-learn, stand-alone, and all-in-one integrated environment.

Through the results of our experiments, we also showed that our visual tools can enhance learners' motivation and performance. In addition an opinion poll showed a positive feedback on the environment tools from the students. In future work, we plan to enhance our visual tools by adding more features, more visual examples and games, and by performing more performance evaluation experiments.

References

1. Bergstrom, H.: Applications, Minimization, and Visualization of Finite State Machines. Master Thesis. Stockholm University (1998), http://www.dsv.su.se/~henrikbe/petc/
2. Bovet, J.: Visual Automata Simulator, a tool for simulating automata and Turing machines. University of San Francisco (2004), http://www.cs.usfca.edu/~jbovet/vas.html
3. Christin, N.: DFApplet, a deterministic finite automata simulator (1998), http://www.sims.berkeley.edu/~christin/dfa/
4. Felder, R., Silverman, L.: Learning and teaching styles in engineering education. Engineering Education 78(7), 674–681 (1988)
5. Hadjerrouit, S.: Learner-centered Web-based Instruction in Software Engineering. IEEE Transactions on Education 48(1), 99–104 (2005)
6. Hamada, M.: Web-based Tools for Active Learning in Information Theory. ACM SIGCSE 38 (2007)
7. Hamada, M.: Visual Tools and Examples to Support Active E-Learning and Motivation with Performance Evaluation. In: Pan, Z., Aylett, R.S., Diener, H., Jin, X., Göbel, S., Li, L. (eds.) Edutainment 2006. LNCS, vol. 3942, pp. 147–155. Springer, Heidelberg (2006)
8. Head, E.: ASSIST: A Simple Simulator for State Transitions. Master Thesis. State Univesity of New York at Binghamton (1998), http://www.cs.binghamton.edu/~software/
9. Herrmann, N.: The Creative Brain. Brain Books, Lake Lure (1990)
10. Java2D of Sun Microsystems, http://www.sun.com
11. Java Team, Buena Vista University, http://sunsite.utk.edu/winners_circle/education/EDUHM01H/applet.html

12. Keller, J.: Development and use of the ARCS model of motivational design. Journal of Instructional Development 10(3), 2–10 (1987)
13. Kolb, D.: Experiential Learning: Experience as the Source of Learning and Development. Prentice-Hall, Englewood Cliffs (1984)
14. Li, S., Challoo, R.: Restructuring an Electric Machinery course with Integrative approach and computer-assisted Teach Methodology. IEEE Transactions on Education 49(1), 16–28 (2006)
15. Masters, J., Madhyastha, T.: Educational Applets for Active Learning in Properties of Electronic Materials. IEEE Transactions on Education 48(1) (2005)
16. Mohri, M., Pereria, F., Riley, M.: AT&T FSM Library Software tools (2003), http://www.research.att.com/sw/tools/fsm/
17. Myers, Gifts Differing. Palo Alto, CA: Consulting Psychologists Press (1980)
18. Nelson, R., Shariful Islam, A.: Mes- A Web-based design tool for microwave engineering. IEEE Transactions on Education 49(1), 67–73 (2006)
19. Rodger, S.: Visual and Interactive tools. Automata Theory tools at Duke University (2006), http://www.cs.duke.edu/~rodger/tools/
20. Rosati, P.: The learning preferences of engineering students from two perspectives. In: Proc. Frontiers in Education, Tempe, AZ, pp. 29–32 (1998)
21. Soloman, B., Felder, R.: Index of Learning Style Questionnaire, http://www.engr.ncsu.edu/learningstyle/ilsweb.html
22. Transforming undergraduate education in science, mathematics, engineering, and technology. In: Committee on Undergraduate Science Education, Center for Science, Mathematics, and Engineering Education. National Research Council ed. Washington, DC: National Academy Press (1999)