An Integrated Virtual Environment for Active and Collaborative e-Learning in Theory of Computation

Mohamed Hamada, Member, IEEE Computer Society

Abstract—Active and collaborative learning provides a powerful mechanism to enhance depth of learning, increase material retention, and get learners involved with the learning process instead of passively participate in it. In this paper, a research using Web-based active and collaborative learning in the theory of computation and related fields is presented. The twofold contribution of this work is a novel use of existing technology to improve learning and a longitudinal quasi-experimental 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, in addition to a support for collaborative learning. For the second contribution, several classroom experiments are carried out. The analysis of the experiments’ outcomes and the students feedback 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.

Index Terms—Computer uses in education, collaborative learning, computer-assisted instructions, distance learning, theory of computation, automata.

1 INTRODUCTION

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 [30]. So far, several learning models have been developed (e.g., [5], [11], [15], and [21]) for the realization of the learning preferences of learners. Among these models, that of Felder and Silverman [5] is simpler and easier to implement through a Web-based quiz system, as in [26]. The model classifies engineering learners into four axes: active versus reflective, sensing versus intuitive, visual versus verbal, and sequential versus 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 [25], 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.

The concepts in the theory of computation course have important use in designing and analyzing computational models of 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.

Our integrated environment (IE) is designed to tackle this issue and meet the active learning preferences for computer engineering learners. IE can be used as a supporting tool for active and collaborative learning not only for the theory of computation course but also for several other courses such as automata and formal languages, discrete mathematics, computational models, principles of programming languages, compiler design, and other related courses. Such courses cover a variety of topics including finite state machines (FSM, automata), pushdown automata, and Turing machines (TMs), in addition to grammars and languages. We cover such important topics in our IE. The covered topics will be written in Java as applets and then integrated into a single environment using the Java2D technology of Sun Microsystems [12]. This implies that our environment is portable, machine independent, and Web-based enabled, which makes it a useful tool as an interactive and online collaborative learning environment.

In designing our IE learning tools, we considered the active construction learning model [8], [27] that has a number of basic design principles, which include the following:

1. Teachers act as facilitators not as knowledge transmitters. This means knowledge must be actively constructed by learners, not passively transmitted by teachers.
2. To motivate learners and get them actively involved in knowledge construction, learning activities should focus around a set of motivating problems and examples that has applications in real world.

3. Learning should take place in a collaborative environment.

4. Assessment procedures should be embedded in the learning process and should consider learners’ individual orientations.

To show the effectiveness of our IE components as a model of interactive online collaborative learning tools, 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.

Despite that we focus on the theory of computation and automata topics, our research can be considered as a model for a wide range of topics in the undergraduate level.

This paper is organized as follows: Following the Introduction, Section 2 discusses some related work. Section 3 overviews the topics covered in our environment, including FSMs, visual machine examples, and TMs. In Section 4, we introduce our IE and all of its components. The learning theory and design behind the development of the environment will be discussed in Section 5. We also will discuss the possibility of the environment integration with the environment will be discussed in Section 5. We also will discuss the possibility of the environment integration with learning management systems (LMSs). The performance evaluation of the environment will be presented in Section 6. Finally, we conclude this paper and discuss the results and possible future extensions in Section 7.

2 RELATED WORK

There are a number of FSM simulators, which have been developed (e.g., [1], [2], [4], [10], [19], and [24]) to enhance the learning of automata 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, the tools PetC in [1] lack visual clarity and dynamic capability. When designing an automaton on PetC editor and try to connect two states in both directions, labels on arrows cannot be distinguished. This becomes visually terrible when the automaton is getting bigger. Moreover zooming in/out to view large automata is not available and TMs are not included. JFLAP [24] 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 [19] are powerful but do not provide a convenient mechanism for displaying and visually simulating the FSMs. The ASSIST automata tools in [10] are difficult to setup and use. Almost all have been designed as tools for advanced learners. These tools assume that the learners have already grasped the fundamental concepts. They lack a clear workflow of learning activities that can guide the new learners how and where to start using the system. This makes it difficult for new students to navigate through the system. They are also dependent on advanced mathematical and idiosyncratic user interactions. On the contrary, our tools are designed with a clear workflow of learning activities, and hence, it is easy to use and easy to learn for new users. In addition, it can be seen as a stand-alone and all-in-one IE. It also provides a unique feature for online communication among learners through the incorporation of an integrated chatting tool. This unique feature is designed to support online collaboration among learners regardless of their location or platforms they are using.

3 THEORY OF COMPUTATION TOPICS

The topics that we will cover in the present version of our environment are FSMs, applications, and TMs. These concepts are the basics of a variety of courses including theory of computations, computational models, discrete mathematics, automata and formal languages, programming languages, and compiler design. In this section, we will give a brief overview of these topics. For a comprehensive coverage of these topics, we refer to [34].

3.1 Finite State Machines

FSM or automata represent a mathematical model for several software and hardware applications. Informally, an FSM is a machine with a finite number of states and a control unit that can change the current machine state to a new state in response to an external effect (input). Depending on the way the machine controller responds to the input, FSMs are classified into deterministic (DFA): if the controller can change from one state to another (one) state, nondeterministic (NFA): if it changes from one state to several states, and nondeterministic with empty move (λ-NFA): if in (addition to NFA) it can also change states in response to empty (no) input.

The three models of FSMs, i.e., DFA, NFA, and λ-NFA, are equivalent. In other words, given any type of the machine, we can transform it into the other. By definition, we can see that DFA ⊆ NFA ⊆ λ-NFA, but we can transform λ-NFA to NFA and NFA to DFA (see Fig. 1).

Regular expressions (REs) are important notations that are not automaton-like but play an important role in the study of automata and their applications. REs can define exactly the same class of languages that the various forms of finite automata can describe. However, REs offer something that automata do not: a declarative way to express the strings we want to accept.

Just as finite automata are used to recognize patterns of strings, REs are used to generate patterns of strings. An RE is an algebraic formula whose value is a pattern consisting of a set of strings, called the language of the expression. REs and FSMs have equivalent expressive power (see Fig. 1).
While teaching these concepts in a traditional lecture-driven style, we noticed that inexperienced (novice) learners have difficulties to fully understand these basic concepts, especially with more complex automata models. Moreover, learners became less motivated to actively participate in the class. To deal with this issue, the FSM component of our IE covers all these concepts in a visual and interactive way that is more suitable for engineering students. Using the editor of the FSM component, learners can easily build, modify, and simulate a DFA, an NFA, or a λ-NFA. Then, they can interactively simulate the machine with any desired input to learn how the machine behaves (in distinct steps or as a whole manner) in response to that input. After building the machine and simulating it with the desired inputs, learners can now perform more complex operations on the machine as easily as clicking a button while learning the underlying algorithms of the performed operations. For example, minimization of a DFA can be performed and visualized in a step-by-step manner to enable the learners to study the minimization algorithm and see how it works on an example. Similarly, the transformation from one type of finite automata to another type or between finite automata and REs (as shown in Fig. 1) are performed with a click of a button and visualized in a way that enables the learners to study the corresponding algorithms in a sequential and visual way.

3.2 Applications

FSMs have several applications in both software and hardware. In software design, it can be used in a wide range of modeling from a simple text editor to a more sophisticated compiler. In computer gaming, it can be used to model puzzles, tennis games, and many others. In hardware design, it can be used to model the function of a variety of machines, for example, vending machines, elevators, video players, rice cookers, and so forth. These important daily life applications must be made clear and visible for learners in order to attract their attention and increase their motivation based on Keller’s ARCS motivation model [14]. To achieve this goal, we designed a set of visual FSM examples as separate components integrated into our environment. In our IE, we added three such examples: a rice cooker, video player, and tennis game simulators. The first two examples show how FSMs can be used to model hardware devices that we use in everyday life. The third example shows how an FSM can be used to model a popular game such as tennis. We also designed more examples such as an elevator, vending machine, and “man, wolf, goat, and cabbage” puzzle, but these examples have not yet been integrated into our environment.

3.3 Turing Machines

TMs are the most powerful FSMs. They can simulate exactly what a digital computer can do. Informally, TM consists of a finite set of states and a controller that can read or write symbols on an infinite length tape. Unlike DFA, NFA, and λ-NFA, a TM controller can move in both directions on the tape. The machine starts with an initial state, a finite number of input symbols written on the tape (all other infinite number of tape cells are blank), and the controller is set to the first input symbol from the left. According to the current state and the current scanned symbol on the input tape, the controller takes the next move. It can overwrite the current scanned symbol or leave it untouched, change the current state, then move to either the left or the right, and so on. If no more moves are possible, then the machine halts. In some cases, the machine may run forever and never halt.

But, more interestingly, TMs can be used to compute functions, exactly the same way that modern digital computers can do. In this case, the function arguments are represented as sequences of 1s separated by 0s and are written on the machine’s tape. The function definition is represented as a set of rules suitable for the machine’s transition function. Then, the machine works on that input. If it halts, the output symbols left on the tape represent the value of the function application on the arguments.

As stated before, traditional lecture-driven style for teaching/learning TMs is time consuming and difficult for average students to grasp its basic concepts. In order to facilitate the teaching/learning of basic TM concepts for average engineering students, a TM simulator component is integrated into the environment. Learners can easily build their own TM and follow in a step-by-step manner how the TM works on any given input. They can build machines that behave as language recognizers in addition to building machines that can behave as function computers. It has a friendly user interface with some animation and sound effects that enhance the component and make learning more attractive, active, and interesting to the learner.

4 Integrated Environment

Our environment contains nine components, which have been integrated into a single unit to make all topics easily accessible for learners. The components include the following: an animated (movie-like) welcome component, a hypertext introduction to the theory of computation topics, an FSM simulator, a TM simulator, self-assessment exercises, a chatting component for supporting online collaborative learning, and the other three components showing the visual examples of FSMs. The welcome and introduction components use plain and animated texts, 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 all the components of our IE.

4.1 Basic Concepts

The first two components of the environment introduce the principle ideas of FSMs. One component presents a short movie-like introduction that welcomes the learners to the topic. The other one is a rich hypertext introduction to the basic concepts. Learners can navigate through the component and learn about the basic concepts. The interfaces of these two components are shown in Figs. 2 and 3. The animated text is combined with an optional audio narration, which is convenient for learners who have difficulties reading text. It is also presented bilingually, in English and Japanese.
4.2 Finite State Machine Simulator

The FSM 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, $\lambda$-NFA to NFA transformation, DFA to RE, and RE to $\lambda$-NFA. This completes the automata cycle transformation shown in Fig. 1. 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. 4.
4.3 Turing Machine Simulator

The TM simulator is integrated into the environment as well. This simulator is based on the work in [13]. 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 \( \text{add}(m, n) = m + n \) is represented by the TM rules shown in Fig. 6. A rule in the form \( ab > c \) 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 \( abd \) 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 \( \text{add}(2, 3) = 2 + 3 \), 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 see how the TM behaves on input and how it can compute functions. All operations of the TMs can be simulated by this simulator. The component interface showing the TM simulator is shown in Fig. 5.

4.4 Visual Examples

In our IE, a set of visual FSM examples are 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 three examples, which we have already integrated into our environment.

4.4.1 Tennis Game

Tennis is one of the popular games worldwide. It is interesting to show to learners that a tennis game could be modeled by automata. It enables them to learn about automata while enjoy playing the game online. 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 autoplay 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. 7 shows a
A snapshot of the tennis game simulator interface. The game is completed when a final state of the 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.

4.4.2 Video Player

The operations of a video player, a common and useful machine in our daily lives, can be modeled by an FSM. Video player operations such as play, pause, forward, fast forward, backward, fast backward, and stop can be represented by the finite automaton states. The video player simulator is designed to play a movie showing a wild cat in response to the change in the finite automaton states. The operations are represented by buttons labeled with real video player symbols. When the user pushes a button, the corresponding automaton state is highlighted and the appropriate screenshot takes place. Fig. 8 shows the video player simulator interface in which the operations of the video player are simulated and the corresponding finite automaton is displayed.

4.4.3 Rice Cooker

Rice cooker operations can also be modeled by an FSM. Examples of the operations include cooking, reheating, and keeping warm. A timer is also considered in this model. The student can operate the rice cooker simulator by pressing the operations, and then the corresponding state of the underlying automaton is highlighted. Fig. 9 shows a snapshot of the rice cooker simulator interface. In the rice cooker automaton model, every state represents an operation, for example, the state labeled $q_0$ represents the waiting (initial) state, $q_1$ represents the keep warm (final state) operation, and $q_2$ represents the reheating operation. The input alphabet is represented by the symbols A, B, C, and D, where

1. “A” corresponds to heating and reheating operations,
2. “B” corresponds to the keep warm and cancel operations,
3. “C” corresponds to the timer, and
4. “D” corresponds to the finish operation.

In a real rice cooker, after we set the initial conditions, it completes the task and finishes automatically after a certain amount of time. The applet simulates such behavior; first, the user can set the initial conditions by pressing the buttons A, B, and C; then, the finish button D will take place automatically by the automaton after a certain amount of time. Thus, we do not need to explicitly add a button “D” to the simulator.

4.4.4 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 TMs, finite automata, or REs. Learners can perform a pre-assessment, an in-assessment, or a post-assessment. The assessment interface is shown in Fig. 10.

First, the learner must select an exercise (upper left corner in the window), 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.

4.4.5 Support for Online Collaborative Learning

Benefits of collaborative learning are clear at hand, among these benefits are:

- developing higher level thinking skills in learners,
- increasing learners retention,
- building self-esteem in learners,
- enhancing learners’ satisfaction with the learning experience, and
- promoting a positive attitude toward the subject matter.

To support online collaborative learning, the chatting component was integrated into the environment. Through this component, learners can communicate online, exchange
questions and answers, and discuss ideas for solving exercises. To use this component, a Java server was installed on a Windows XP machine. The Java server requires the open source free software Apache [28] and Tomcat 5.5 [29] software. Fig. 11 shows a snapshot of the chatting component. The figure shows three online learners discussing the answer to questions from the exam components.

Many learners used the chatting component effectively to solve assignments and to discuss new ideas.

5 LEARNING THEORY

There is a limited value in asserting which theory of learning is better. Rather, an integrative approach should be
preferred. One should be able to combine different instructional conditions necessary to promote effectively a given type of learner performance. From the content-design point of view, one can glean the following important issues from the many instruction-design theories proposed in literature [6], [18], [20], [23]:

- knowledge classification;
- granularity of learning unit;
- content sequencing and adaptive navigation through the instructional environment; and
- learning events as per the required learning outcome.

Our IE was designed with these principles in mind. In the following sections, more clarification will be given on the design process of IE.

5.1 Learning Design

The central ideas behind learning design represent new possibilities for increasing the quality and variety of teaching and learning within an e-learning context [3]:

- The first general idea behind learning design is that people learn better when actively involved in doing something (i.e., are engaged in a learning activity).
- The second idea is that learning activities may be sequenced or otherwise structured carefully and deliberately in a learning workflow to promote more effective learning.
- The third idea is that it would be useful to be able to record "learning designs" for sharing and reuse in the future.

Following the IMS-LD learning design specifications [31], our IE is build up on the following sets of objects and activities.

**Objects.** The learning objects are the environment components described as follows:

1. The “Topics” object, which is a set of URLs, contains a description of the course topics. Learners can navigate these topics in a sequential manner or by a selection of a specific topic from the table of contents.
2. A set of motivating examples reflecting the use of the topics in daily life activities such as using vending machines, elevators, video players, tennis game, rice cookers, and solving puzzles. Learners can play with a simulation of these activities and learn how they are related to the theory of computation topics.
3. A simulator of FSMs. FSMs are mathematical models for many software and hardware applications. The simulator object enables the learners to build their machine and test how it reacts in response to different inputs. It also enables learners to test the theoretical concepts they study in the “Topics” object.
4. A TM simulator. TMs are more powerful machines than FSMs. TMs can be considered as a theoretical model of modern digital computers. The TM simulator object enables the learners to build their machine and test it with different operations such as computing functions. This can deepen the learners understanding of the theoretical part of the course topics.
5. A set of tests for self-assessment. Tests go from easy and general to difficult and more specific. For self-assessment, learners can try tests at any stage of the learning process.
6. A chatting object as a support for collaborative learning. Two or more learners can chat together at any stage of the learning process to exchange opinions and ideas.

**Activities.** A recommended set of activities that learners can consider when using the environment is described in the following sequential order:
1. Learners are recommended to start using the environment by playing with the visual examples. This does not need any special knowledge or background of the topics. It also will attract the learners’ attention to the relevance of the topics. Learners’ attention and topic relevance are the basics of Keller’s ARCS motivation model [14].

2. Taking the first simple (general) test. By answering the easy and general questions in this test, the learner will gain self-confidence, which is an important factor of learners’ motivation in ARCS motivational model [14]. At this stage, learners are ready to start reading the theoretical concepts in the topics object.

3. Navigating the concepts in the topics object will give the learners the necessary theoretical background for the subject.

4. By now, learners are ready to use the FSM and the TM simulators. Switching between reading the topics and using the simulators is recommended. After reading a certain topic, the learner can switch to the simulator and try to build a model for that topic and test the model with different inputs. This can help in deepening the learners’ knowledge and enhances the learning process.

5. During reading the topics and using the simulators, learners are recommended to try the corresponding test (in the test object) for self-assessment and to gain more confidence about their learning progress.

6. At any stage of the learning process, online learners can chat with each other through the chatting object. This enables learners to exchange ideas and help each other to understand the topics and answer the test questions in a collaborative way.

The environment objects and the workflow of the learning activities within the environment are shown in Fig. 12.

5.2 Learning Management Systems

An LMS is a software application or Web-based technology used to plan, implement, and assess a specific learning process. Typically, an LMS provides an instructor with a way to create and deliver content, monitor student participation, and assess student performance. An LMS may also provide students with the ability to use interactive features such as threaded discussions, video conferencing, and discussion forums. The Advanced Distance Learning group [28], sponsored by the United States Department of Defense, has created a set of specifications called Shareable Content Object Reference Model (SCORM) to encourage the standardization of LMSs. Another well-known model of LMSs is the Modular Object Oriented Dynamic Learning Environment (MOODLE) [33]. MOODLE is a course management system, which has been designed using sound pedagogical principles. It supports a range of different resource types that allow including almost any kind of digital content into courses. The resource may already exist in electronic form so it can be linked to an uploaded file or external website or simply displaying the complete contents of a directory in the course files and then users can pick the file themselves.

Since our IE is based on a collection of HTML materials and Java applets, its integration with LMS such as MOODLE can be easily achieved. While we believe that our IE is currently sufficient for achieving its goal, it is worth trying its integration with well-known LMSs such as MOODLE in the near future. This may enhance the IE functionality by adding new operations such as video conferencing and gives the teachers a chance to track the learners’ activities in IE. As for integration with SCORM, there is nothing preventing the Java applets of IE from being used in either a content package or an LMS. SCORM does not specify implementation details such as which languages can be used in the development of content packages or LMSs, other than the fact that SCORM must be able to communicate with the LMS’s API via Java script.

6 Evaluation

We carried out four experiments in order to evaluate the effectiveness of our IE tools on the learning process of engineering students. The first experiment evaluates the learning preferences of the students according to the “Learning Style Index” of Soloman and Felder [26]. The second experiment evaluates the improvement in the students’ motivation based on Keller’s ARCS motivational model [14]. The third experiment (statistically) evaluates the effectiveness of using the tools on the students’ performance. Finally, an opinion poll was carried out in order to observe the students’ feedback about the tools.

To help learners find their learning preferences, the Felder-Soloman Index of Learning Style [26] was introduced. Fig. 13 shows a summary of the learning style quiz results from the author’s evaluation as well as the data found at the University of Western Ontario by Rosati [25] where he surveyed 800 students and found that engineering students typically have preferences toward active, visual, and sensing learning preferences. It also contains a similar data by Masters and Madhyastha at San Jose
University [17]. We found that our data support Masters and Rosati's data.

The purpose of introducing the visual automata examples in IE is to enhance the students' motivation. Our evaluation of motivation is based on “Attention” and “Relevance,” which are the basics of Keller’s ARCS motivation model [14]. 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: “don’t know,” “no,” “maybe no,” “maybe yes,” and “yes.” The responses are shown in Table 1 (Before). Then, we demonstrated our visual examples to the students and repeated the same question again. Their responses (after seeing the examples) are shown in Table 1 (After). Comparing the results from “Before” and “After” exposures to the examples, there was a slight improvement in motivation. For choices “don’t know,” “no,” and “maybe no,” if the number of responses decreased, it indicates a positive result, which is what occurred. While for the other choices “maybe yes” and “yes,” the increasing number of responses indicates positive result, which also occurred. These results are summarized in Fig. 14.

We note that there was only a slight 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 previous 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: “don’t know,” “no,” “maybe no,” “maybe yes,” and “yes.” Their responses (before seeing the examples) are shown in Table 2 (Before). Next, we demonstrated our visual examples to the students and presented the same question to them again. Their responses (after seeing the examples) are shown in Table 2 (After). Comparing the results “Before” and “After” exposures to the examples, we can see a better improvement in their motivation. As with the previous experiment, for choices “don’t know,” “no,” and “maybe no,” if the number of responses decreased it meant a positive result, which is what occurred. While for the other choices “maybe yes” and “yes,” an increasing number of responses meant a positive result, which also occurred. These results are summarized in Fig. 15.

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. We found that the visual examples attracted the students’ “attention” to the subject and showed its “relevance” in practice. “Attention” and “relevance” are the basics of Keller’s ARCS motivation model [14]. The increase in the students’ motivation was reflected in a significant increase in class participation and general interest in the subject.

A preliminary study shows that the IE can improve the learning process of computer engineering students who study the theory of computation course and related courses. Last semester, the students were divided into four groups, each group containing 20 students. A set of 40 (sufficiently complex) exercises was distributed among the groups, 10 for each group. For example, a construction of FSMs that simulate a complex vending machine (or other kind of machines). Each exercise consists of the following:

1. Planning the construction of the machine: how many states are needed, which input set is required, what is the set of outputs, and so forth.
2. Designing the machine: connecting the states and define the move function.
3. Test the machine with different inputs and check the corresponding output.
4. If the machine is NFA (nondeterministic), covert to a DFA (deterministic) machine.
5. Optimization/minimization: apply a minimization algorithm to get an optimized machine.

Each of the 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 IE and the other two groups without using it. An equal time period (between 1 and 3 hours: depending on the complexity of the exercises) was provided to all the groups. The result showed a better performance for the two groups using the IE: solving more exercises in a shorter time. Then, the experiment was repeated by redistributing the exercises among the four groups. Again, the two groups with IE showed better performance. In the average, the students using IE could answer 85 percent of the questions and the students without it could answer 55 percent of the questions.

![Fig. 15. Improvement in the junior learners’ motivation after demonstrating our visual examples.](image-url)
To also show how the student performance was improved by using IE, we survey the students’ grades for the last four years, as shown in Table 3. These results corroborate the positive effects of applying these educational tools in automata course. The evaluation was based on exam, exercises, and several laboratories. The first two years, when IE had not yet been used, a considerable number of students failed (F) the course. The third year, when the first version of IE was introduced, the number of failure was reduced and the number of high scores (A and B) was increased. In the forth year, when the improved IE (with more functionality) was introduced, a better performance was obtained: more reduction in failure and more increase in high scores.

In addition to the experiments, an opinion poll among the learners was carried out. The result of the poll is shown in Table 4. Among the 80 learners in the class, 78 had completed the poll. As shown in Table 4a, among the 78 responses, 71 percent preferred using the environment tools as shown in Table 4b. Most questions on the opinion poll were Likert-type questions that made a clearly negative or positive statement about the IE tools and allowed the learners to strongly agree, agree, be uncertain, disagree, or strongly disagree. Scores for the environment tools were generated based upon the learner responses. The scores could fall between 10 (worst) and 50 (best) and were divided into five ranges: greatly disliked the tools (score: 0-10), disliked the tools (score: 11-20), uncertain in preferences for the tools (score: 21-30), liked the tools (score: 31-40), and greatly liked the tools (score: 41-50). Table 4c shows that the average score for the environment tools lies on the far end of the “liked the tools” range.

Tables 4d and 4e show the responses to other important questions. These results show that the majority of learners found that the environment tools helped clarify important concepts and encouraged them to think about concepts outside of class. The latter is a significant accomplishment that could lead learners to seek more knowledge and information on their own.

### 7 Conclusion

With the vast advancement in technology, the traditional lecture-driven classroom is giving way to a new and more active environment, where students have access to a variety of multimedia and interactive course materials. Such interactive course materials have already been introduced for several topics in engineering courses; see for example [7], [9], [16], [17], and [22].

In this paper, we have followed the same path and introduced an environment that integrates a set of visual tools to support interactive and collaborative learning in the theory of computation course. It can also be used in other courses such as model of computations, language processing, automata and formal languages, compiler design, discrete mathematics, and other similar courses. 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 and more visual examples, and by performing more performance evaluation experiments specifically, we need to measure the time required for topic learning with and without the tools. We are also developing a mobile version of the tools that can run on devices with limited capabilities such as mobile phones and personal digital assistants (PDAs), which are increasingly popular among students.

The experiments and the opinion poll showed that our IE and its components are useful tools for supporting collaborative online e-learning in a variety of courses. Our tools are Web based, easy to use, all in one, and stand alone, which make it a useful tool of e-learning and can also be integrated into LMSs such as MOODLE.

### REFERENCES

Mohamed Hamada received the BSc and MSc degrees (with honors) in pure mathematics and computer science from Ain Shams University, Cairo, and the PhD degree in computer science from the University of Tsukuba, Japan. He is currently an assistant professor in the Language Processing Systems Laboratory, Software Department, University of Aizu, Aizuwakamatsu, Japan. His research interests are in advanced learning technologies, naturally inspired computations, and programming languages. He is a member of the IEEE Computer Society and the ACM.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.