A kinesthetic-based collaborative learning system for distributed algorithms

Hiroyuki Nagataki[*], Taichi Fujii[†], Yukiko Yamauchi[‡], Hirotsugu Kakugawa[†] and Toshimitsu Masuzawa[†]

[*]Center for Faculty Development, Okayama University, Okayama, Japan
Email: nagataki@cc.okayama-u.ac.jp

[†]Graduate School of Information Science and Technology, Osaka University, Osaka, Japan
Email: {t-fujii, kakugawa, masuzawa}@ist.osaka-u.ac.jp

[‡]Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan
Email: y-yamauchi@is.naist.jp

Abstract—In this paper, we present a learning support system DASE-E to help students understand fundamental concepts of distributed algorithms in computer science. DASE-E is a collaborative learning system, in which the task of students is to devise a distributed algorithm. DASE-E offers a set of small wireless terminals with accelerometers. Each student plays the role of a process with a terminal, according to the algorithm that students devised. Each terminal enables a student to take physical actions that control the behavior of a process in the simulator. After the role playing simulation is finished, students discuss their activity played back on a screen.

We implemented the system for learning distributed leader election algorithms, had a trial exercise in our research group, and we confirmed that it is effective to learn the critical part of distributed systems and algorithms.

Keywords-computer science education, distributed algorithm, collaborative learning, kinesthetic learning activity, algorithm visualization

I. INTRODUCTION

A distributed system is a network which consists of a set of processes (*i.e.*, computers or programs) that communicate with each other by exchanging messages. A distributed algorithm defines the procedure at each process to perform a task in a distributed system. A distributed system models computer networks such as the Internet, P2P networks, mobile ad-hoc networks, and sensor networks.

As computer networks become widely used, learning and teaching distributed algorithms become more important. For example, in CS2008 [1] distributed algorithm is one of the important learning topics for undergraduate computer science education. Through a class, students are required to understand the fundamental concepts behind distributed algorithms: how a process operates, why the algorithm works, what is the critical part of the algorithm, etc.

However, it is difficult for students to learn the fundamental notions in distributed computing, because it has many complicated characteristics. One of the difficulties is the nondeterminism of execution. Each process executes the algorithm asynchronously, concurrently and at a different processing speed, and then changes its state and exchanges messages with other processes. Messages are also exchanged

concurrently and their delivery delays differ at times. Hence, numerous numbers of executions are possible because of the nondeterminism, and students cannot imagine all cases of executions, especially, some critical executions for an algorithm. Another difficulty is that each process executes the algorithm with limited information about the network, because each process can communicate with only its neighboring processes.

Many pedagogical researches indicate that *collaborative learning*, based on the concept of social constructivism, is more effective for learning than only listening lectures or viewing animation. Also, *kinesthetic learning* has become one of the popular methods in computer science education [2].

Based on these two learning methods, we present a learning support system **DASE-E** (Distributed Algorithm Simulator Engine - for Education), which supports collaborative learning activity of distributed algorithms. DASE-E is designed to help students understand the fundamental concepts of a distributed algorithm by a simulation activity, in which each student plays a role of each process in a distributed system. DASE-E offers a basic environment for distributed algorithm simulation, though the behaviors of processes are controlled by the students' kinesthetic activity. Students can easily devise and try to simulate various distributed algorithms on DASE-E, and discuss among students to evaluate their algorithm. Learning with kinesthetic activity offered by DASE-E will help students understand the fundamental concepts of distributed algorithms.

II. RELATED WORK

There are many approaches proposed to support learning or teaching distributed algorithms. Common observation indicated by these studies is that understanding the fundamental concepts of distributed algorithms is difficult for students, and to achieve it, a tool or method is needed to let them observe actual behavior of algorithms.

One of the basic approaches for learning distributed algorithms is to make students implement an algorithm. Through the implementation process and executing programs, they review and observe how the algorithm is designed and executed. However, this approach has some drawbacks. Students tend to struggle with coding and removing subtle

bugs that are not related to understanding the algorithm. Hence, many learning systems for algorithm implementations propose frameworks that help implementation of distributed algorithms [3].

Algorithm visualization is also widely used for learning or teaching sequential algorithms [4]. This approach is also adopted in some learning systems of distributed algorithms [5]. However, visualization of distributed algorithms has some difficulties. In a distributed system, processes execute the algorithm asynchronously and concurrently. It is difficult for students to observe what is going on in the distributed system with naive visualization, which will display a large amount of information: concurrent and continuous state changes at processes and message transmission.

Several algorithm visualization systems improve the above drawbacks by providing interactive algorithm simulation [6] [7] [8], in such a way that students can control the execution of a distributed algorithm by a script or real-time operation through GUI. They support operations to pause, skip, rewind, and repeat an execution, with which students can observe the behavior of processes and links whenever they want. However, this approach is not suitable for students, who do not understand the fundamental idea of the distributed algorithm. Without sufficient knowledge about the algorithm, it is almost impossible to recognize when, where and what they should focus on in the simulation to understand the algorithm.

On the other hand, *kinesthetic learning activity* is proposed from pedagogical research. Kinesthetic learning activity is a pedagogical style involving physical actions by students. This approach is also proposed for distributed algorithm education [9]. In [9], they showed two important points: kinesthetic learning activity is also an effective pedagogical style for computer science education, and collaborative work is an effective method for simulating distributed algorithms, which makes it easy for students to understand concurrency and locality of a distributed algorithm by playing a role of a process. However, this approach has a drawback. In this activity, each student only knows his/her local activity. It is difficult for the student to evaluate the entire activity of the distributed system.

Our approach is to integrate algorithm visualization and kinesthetic learning activity that are complementary to each other. By role-playing in a kinesthetic learning activity, our approach enables each student to understand locality and concurrency of a distributed algorithm from a local point of view. In addition, by algorithm visualization that plays back students' activity, our approach enables each student to see the entire behavior of a distributed system from a global point of view.

III. MAIN CONCEPT OF DASE-E

In this section, we present the main concept of DASE-E. First, we introduce a scenario of a learning activity with DASE-E, and based on the scenario, we discuss system requirements for DASE-E.

A. Learning Scenario

We consider the following scenario in a class of distributed algorithms. The goal of the class is to understand distributed computing by devising a distributed algorithm.

First, a teacher gives a lecture to students about computational models and characteristics of a distributed system, for example processes, message passing, nondeterminism, and concurrency.

Next, the teacher gives students a project issue for distributed algorithms, for example "Design an algorithm to elect a leader process in the network" (*i.e.,* the leader election problem). The teacher explains details of the problem setting such as network topology, the number of processes, the information each process initially knows, and so on.

Students are divided into several groups, and DASE-E is given to each group. In each group, students devise an algorithm for the problem through a group discussion. And they simulate the algorithm with DASE-E.

In the simulation, each student operates a process in a distributed system according to the devised algorithm. DASE-E offers a number of small wireless terminals. Each of them corresponds to a process in a distributed system. Each student has a terminal, and the terminal displays the local state of the corresponding process so that he/she can check it at any time.

When a student takes a physical action with the terminal, a process takes a corresponding action, which changes the local state of the process or sends/receives messages. During the simulation, students are not allowed to talk to each other, and they only look at and operate the terminal. Each student only knows the local state of the corresponding process during the simulation.

After the simulation is finished, students evaluate whether their algorithm was successful or not. DASE-E provides a visualization window on PC, which replays the execution of the distributed system the group has just simulated. Students observe the activity of processes on the screen, and they discuss the correctness and the efficiency of the algorithm. If students find that their algorithm seems not good (*e.g.*, the number of messages is too high), they devise another algorithm. They repeat their activity until they achieve the best algorithm: devising an algorithm, simulation, observation and evaluation.

Finally, the teacher evaluates the algorithm each group devised, and shows an example of a well-known distributed algorithm that teacher wants to teach students in the class. Students simulate the algorithm with DASE-E, and discuss the algorithm taught by the teacher and the algorithm they devised.

B. System Design

DASE-E should provide two modes, the *interactive simulation mode* and the *playback mode*.

In the interactive simulation mode, DASE-E simulates a distributed system, in which each process is controlled by the user. Each student uses a terminal to control a process. The terminal senses physical actions of the student, and converts them into the operations of the corresponding process.
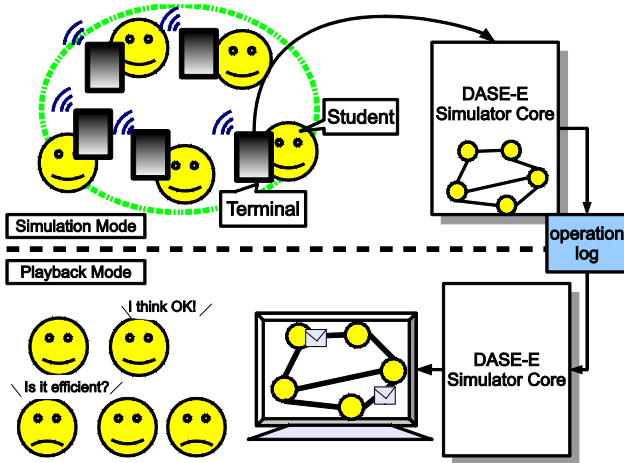
Figure 1. Overview of DASE-E

Figure 2. An example visualization of playback mode

In the playback mode, the system plays back the execution performed in the interactive simulation mode on PC display. For the playback, DASE-E records the operations by students in the interactive simulation mode.

Now, we define the requirements for developing DASE-E as follows:

1. It has an interactive distributed algorithm simulator, which has functionality that each process is controlled by a student.
2. As a control device of the simulation, it uses a terminal with sensors that can sense the physical actions of a student.
3. It has the recording function that records the sequence of all user operations.
4. It has a visualization that displays the execution of the distributed system with GUI. It plays back the activity which has been done in the interactive simulation mode.

IV. ARCHITECTURE

DASE-E consists of one program and a number of terminals. The program is a distributed algorithm simulator running on a PC with a visualization tool of algorithm executions. Each terminal has the sensor that detects students' physical actions. Students can control the behavior of processes in the simulation through the terminals. We assume that one person controls one terminal, so that the number of processes is equal to the number of terminals.

DASE-E has two modes. In the simulation mode, each student uses a terminal to control the behavior of a process (see Fig. 1). The operations of each terminal are recorded in the log file with the times the operations have occurred. After the simulation mode is finished, DASE-E executes the playback mode. In the playback mode, DASE-E reads the operation log and replays the execution that users operated. The distributed system is visualized on the screen with an abstracted graph model (see Fig. 2).

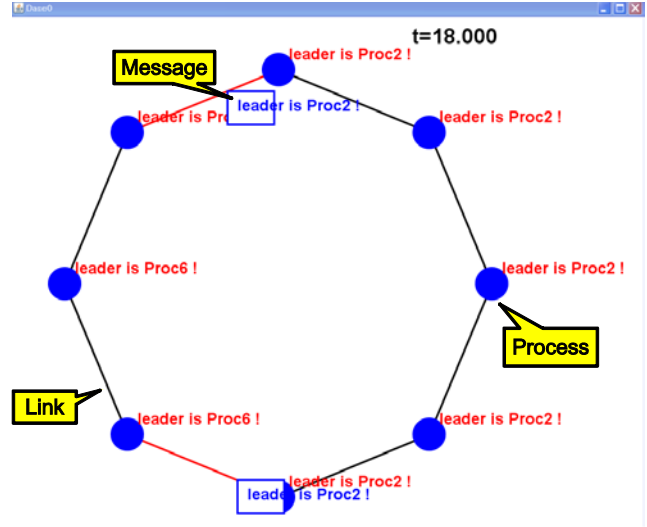Simulator Core in Fig. 1 is the main module for running simulations. It seems similar to other distributed algorithm simulators presented in section II. However, DASE-E is different from them in the behavior of processes; all operations of each process are controlled by a user outside of the simulator.

Simulator Core has some templates of objects such as processes, links, messages, and so on, which are the components of a distributed system (see Fig. 3). In the beginning of a simulation, the simulator creates instances of these templates and runs them in the simulation. Each process instance has a set of local variables that store any type of values, and basic functions of process actions such as sending or receiving a message or changing the values of local variables. These functions are customizable according to the problem. However, no function in the process is executed automatically. Each function is executed only when a trigger signal comes from the outside of the Simulator Core.

The dataflow from the terminal operation to the process in the Simulator Core is shown in Fig. 4. In DASE-E, a control signal from a terminal doesn't reach a process instance in the Simulator Core directly. When a user operates a terminal, the signal is first sent to the DeviceConverter. DeviceConverter is a device-dependent module constructed for each device type, to convert the raw signal of the device into a respective *physical-activity* signal. For example, continuous change of acceleration sensed by a terminal is converted by DeviceConverter into a signal "shake". Then MessageConverter receives the converted signal. MessageConverter is a device-independent but problem-dependent module, which converts a physical-activity signal into a respective action trigger that raises the execution of the process. For example, a MessageConverter module converts the "shake" operation into the trigger "send a message". These modules also support reverse conversion, such as information of a process state into an output signal of the respective terminal, for example blinking LEDs on the terminal.
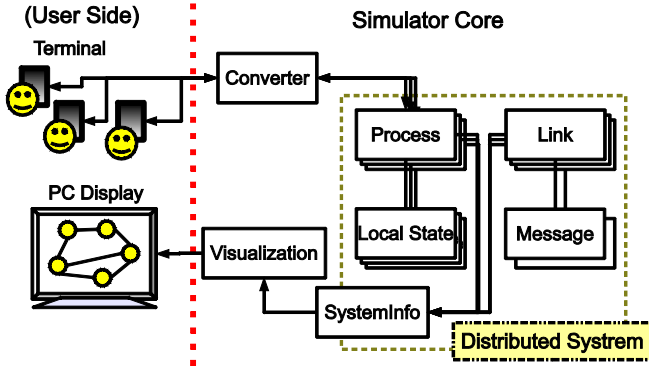
Figure 3. Internal structure of DASE-E Simulator Core

Figure 4. The dataflow of a user's operation

V. IMPLEMENTATION

In this section, we will explain an implementation of DASE-E for the leader election problem as an example. The leader election problem is to elect one process in a network as a leader and to make other processes recognize the leader.

In this implementation, network topology is limited to a unidirectional ring network. It is because we want to let students concentrate on the behavior of processes. Moreover, typical leader-election algorithms presented in textbooks of distributed algorithms (*e.g.*, [10]) assume unidirectional ring networks. In such algorithms, each process has a unique identification number (ID), and the process with the smallest ID is elected as the leader. Hence, in this implementation, each terminal has a unique ID so that the students can simulate typical algorithms in addition to algorithms they devised.

We chose a SunSPOT [11] as a terminal of DASE-E. SunSPOT is a small terminal for sensor networks, which has an accelerometer, a light sensor, and a temperature sensor. It also has two push buttons as input, and eight multicolored LEDs as output. Nevertheless, we have designed DASE-E in a device-independent manner, so that another sensing terminal can be easily incorporated.

The student is responsible for managing messages, and remembering received ID and the local state of the process, which is a part of operations the process is supposed to do. In this implementation, students are allowed two operations of a process: 'send a message to neighbor processes' or 'discard the received message'. Students can also check a process's own ID and the latest received ID on SunSPOT terminal at any time. However, the process doesn't offer memory space to a student, so that he/she should memorize in his/her head or write down the value displayed on LEDs if necessary.

SunSPOT senses the acceleration of its movement, which is converted to the sending/receiving messages and local computation at the corresponding process. If a student shakes a SunSPOT from side to side, it means 'send a message to neighbor processes', and if he/she shakes it back and forth, it means 'discard the received message'. This definition aims to provide intuitive operation: the student operates the device as if the device is a received message itself. Students can send a message only when a message comes from the
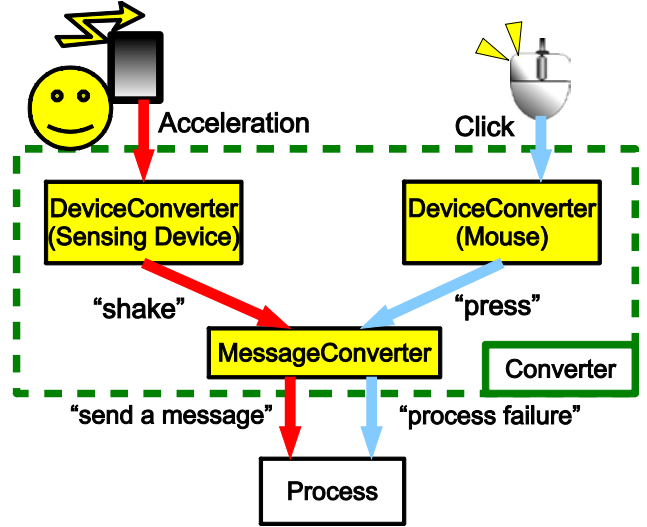
neighbor. It is a fundamental condition of the message passing model, which this implementation supposes, and is also a hint for students to devise an appropriate distributed algorithm. However, of course, this manner is easily customizable according to the problem.

LEDs on SunSPOT display the process's own ID number or the latest number received from its neighbor process. LEDs show the number with four colors; red, green, blue and white, each of which means 50, 10, 5 and 1, respectively. For example, two reds, one blue, three whites means 113. The readability of this indication is better than a binary indication.

Experiment: We used the system in an experimental learning environment. In this experiment, one participant acted as a teacher and five participants as a group of students. They played the scenario described in section III.A. After the experiment was finished, we interviewed participants about the usability of the system. They evaluated that this system is effective especially for experiencing the difficulty caused by concurrency and locality of the distributed system. On the other hand, there were opinions that we need to review the operation method of a terminal, because two types of shake operations caused several wrong operations in the experiment.

VI. CONCLUSION AND FUTURE WORK

We presented DASE-E, a learning support system for distributed algorithm education. DASE-E consists of Simulator Core program and a number of sensing terminals, with which each student can execute an algorithm via physical actions. This system enables students to join the kinesthetic learning activity of distributed systems and supports collaborative work to understand fundamental concepts of distributed algorithms. We also presented an implementation of DASE-E that enables the simulation exercise of leader election algorithms.

DASE-E has to support various algorithms, such as consensus, replication, spanning-tree construction, and so on.

However, customizing DASE-E is a hard work for teachers. Hence, we are now developing a new feature on DASE-E that is easy to apply various distributed algorithm exercises. The feature includes a GUI interface and an easy-writable definition language to define situations of a distributed algorithm.

Moreover, we will evaluate which physical actions are suitable for the simulation of distributed algorithms, and which devices are suitable to sense such actions.

REFERENCES

[1] ACM, IEEE Computer Society, "Computer Science Curriculum 2008: An Interim Revision of CS 2001," ACM (online), available from http://www.acm.org//education/curricula/ComputerScience2008.pdf, (accessed 2010-01-18).

[2] T. Bell, I. H. Witten, and M. Fellows, "Computer science unplugged," http://csunplugged.com/.

[3] R. Oechsle and T. Gottwald, "DisASTer (Distributed Algorithms Simulation Terrain): A Platform for the Implementation of Distributed Algorithms," in ITiCSE '05: Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education.1em plus 0.5em minus 0.4emACM, Jun 2005, pp. 44–48.

[4] M. Krebs, T. Lauer, T. Ottmann, and S. Trahasch, "Student-built algorithm visualizations for assessment: Flexible generation, feedback and grading," in ITiCSE '05: Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education.1em plus 0.5em minus 0.4emACM, Jun 2005, pp. 281–285.

[5] Y. Moses, Z. Polunsky, A. Tal, and L. Ulitsky, "Algorithm visualization for distributed environments," Journal of Visual Languages & Computing, vol. 15, no. 1, pp. 97–123, Sep 2003.

[6] M. Ben-Ari, "Interactive execution of distributed algorithms," ACM Journal of Educational Resources in Computing, vol. 1, no. 2, pp. 2–8, Jun 2001.

[7] S. Carr, C. Fang, T. Jozwowski, J. Mayo, and C. kuang Shene, "Concurrentmentor: A visualization system for distributed programming education," in Proceedings of the 2003 International Conference on Parallel and Distributed Processing Techniques and Applications.1em plus 0.5em minus 0.4emCSREA Press, 2003, pp. 1676–1682.

[8] B. Koldehofe, M. Papatriantafilou, and P. Tsigas, "LYDIAN: An Extensible Educational Animation Environment for Distributed Algorithms," ACM Journal on Educational Resource in Computing, vol. 6, no. 2, pp. 1–21, Jun 2006.

[9] P. A. G. Sivilotti and S. M. Pike, "The suitability of kinesthetic learning activities for teaching distributed algorithms," in SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education.1em plus 0.5em minus 0.4emACM, Mar 2007, pp. 362–366.

[10] N. A. Lynch, Distributed Algorithms.1em plus 0.5em minus 0.4emMorgan Kaufmann, 1996.

[11] Sun Microsystems, Inc., "SunSPOTWORLD," http://www.sunspotworld.com/.