A Proposal for UI-flexible, Loosely-coupled Programming Learning System for Undergraduates

Tomokazu Hayakawa, Chika Nishikado, and Teruo Hikita

***Abstract*—As the scale and the complexity of computer systems increase, the importance of programming education in universities enlarges in these days. In this paper, to improve the quality of programming education in universities, we propose a programming learning system for undergraduates who learn programming. Our proposed system provides a similar experience to pair programming by using static code analyzers as teachers, which means that the system can teach undergraduates many aspects of programming. We designed the system to have UI (User Interface) flexibility and to be loosely-coupled by using REST (Representational State Transfer) in order to increase the maintainability of the system. We implemented the system as an SPA (Single-page Web Application) in order to increase the interoperability between the system and LMSs (Learning Management Systems). We evaluated the system and conclude that the system is a great help for such undergraduates.**

Keywords—e-learning; programming; web service; REST; SPA

I. INTRODUCTION

As the scale and the complexity of computer systems increase, the importance of programming education in universities enlarges in these days. Nowadays undergraduates who are going to be a computer engineer are required to understand many aspects of programming to realize robust and reliable computer systems. For example, it is essential for better design and implementation to understand not only "algorithms and data structures" but also detailed knowledge of programming languages. This is because each programming language has different characteristics, and sometimes a lack of language-specific knowledge causes serious bugs.

As a programming education technique, pair programming plays an important role. In pair programming, each trainee (an undergraduate, in case of universities) has his/her own trainer (a teacher, in case of universities), and they develop software together on a shared PC. Although pair programming spends more time than individual programming, pair programming reduces the number of defects than individual programming.

We believe that pair programming plays an important role in universities to teach undergraduates many aspects of programming. This is because each undergraduate's programming understanding tends to vary considerably; someone struggles with fundamental structures of a programming language such as `if` and `for`, while another one has a difficult time with recursive calls of a function. If each undergraduate had his/her own teacher, he/she could be taught by the teacher that some potential problems exist in his/her source codes that seem to be correct.

However, we consider that it is difficult to do pair programming in most universities owing to lack of the number of teachers. Although pair programming requires as many trainers (teachers) as trainees (undergraduates), an ordinary university class consists of a few teachers and a large number of undergraduates. To do pair programming in universities, we need a solution to compensate the lack.

Generally, to compensate the lack of the number of teachers and to improve the quality of education in universities, using LMSs (Learning Management Systems), such as Moodle [1] and Sakai [2], can be a solution. This is because LMSs can provide a wide variety of educational contents, and undergraduates can learn from the contents at any time on LMSs. As we already know, using an LMS is a good idea in most cases.

Nevertheless, we consider that using LMSs cannot be a great help for programming education owing to the reasons discussed in Section II-A. In short, most LMSs intend to be used for general education, not for programming education.

In this paper, to improve the quality of programming education and to compensate the lack of the number of teachers in universities, we propose a programming learning system for undergraduates who learn programming. (Since we have already published a brief concept of the system [3, 4], we describe the details of the system in this paper.) The system provides a similar experience to pair programming for undergraduates by using static code analyzers as teachers (as discussed in Section II-B), meaning that undergraduates can learn many aspects of programming from the system, not from a human teacher. We designed the system to have flexible UIs (User Interfaces) and to be loosely-coupled by using REST (Representational State Transfer) in order to increase the maintainability of the system. We implemented the system as an SPA (Single-page Web Application) in order to increase the interoperability between the system and LMSs. We evaluated the system and conclude that the system is a great help for undergraduates who learn programming.

This paper is organized as follows. Section 2 introduces related work. Section 3 describes our proposed system. Section 4 and 5 give design and implementation of the system, respectively. Section 6 shows results of our evaluation. Section 7 presents our conclusion.

Computer Science Dept., School of Science and Technology, Meiji University, 214-8571, Kawasaki, Japan

DOI: 10.5176/2251-3043_4.4.343

II. Related Work

A. LMS (Learning Management System)

LMSs (Learning Management Systems), such as Moodle and Sakai, are some kind of Web applications that can provide a wide variety of educational contents for their users, thereby increasing the quality of education.

However, it is difficult to use LMSs for programming education because of the following three reasons. (1) Most LMSs intend to be used for general education, not for programming education. This means that they lack functionalities for programing education. (2) Although most LMSs provide APIs (Application Programming Interfaces) to develop their plugins/extensions to extend their functionalities, developing an LMS-dependent plugin for programming education causes another issue. The developer of such a plugin/extension is required to maintain the plugin/extension whenever the underlying LMS becomes updated or obsolete; otherwise the plugin/extension can no longer be used in the near future. (3) Most existing LMS-dependent programming education plugins/extensions seem to have a fixed UI. However, we consider that such a fixed UI is not suitable for programming education as discussed in Section III-B-1.

B. Static Code Analyzers

Static code analyzers are tools for checking source codes to identify potential problems. For Java, for example, there are a large number of static code analyzers such as CheckStyle [5], FindBugs [6], and PMD [7]. They are already widely used in practice to increase the quality of source codes.

We have decided to use static code analyzers as teachers in our proposed system. Although static code analyzers may not be sufficient as teachers in comparison with human teachers, we presume that static code analyzers can be of great help because of the fact that they are already widely used in practice.

C. Online Web Sites

There are some Web sites, such as codepad.org [8] and Ideone.com [9], to write a source code and execute it on a Web browser. Such Web services and our proposed system are similar in that both provide online editing and online execution functionalities. However, such Web services are not designed for programming education, lacking functionalities to identify potential problems in submitted source codes.

VirtualPairProgrammers.com [10] is a Web site that provides video training courses to learn Java. The Web site claims that the video training courses are the best way to learn new Java programming skills. The Web site and our proposed system are similar in that both consider pair programming as the effective method for programming education. However, the Web site and our system are different in how to provide virtual pair programming environment. The Web site uses videos, which are one-way communication from teachers to students, to provide educational information. On the other hand, our system uses static code analyzers as teachers to realize two-way communication; when a user of our system changes his/her source codes, newly detected potential problems are immediately pointed out, if any, on the screen.

D. Related Studies

Although there are some researches that are somewhat similar to ours, few researches aim to improve each undergraduate's programming understanding by using static code analyzers without heavily depending on LMSs. Jelemenska et al. [11] propose a system that provides SystemC functionalities on Moodle. It is different from our proposed system in that their proposed system is focusing on SystemC and it can run on Moodle only. Itou et al. [12] propose a method that uses Web services to increase functionalities of LMSs. Their research and ours are similar in that both aim to realize a loosely-coupled architecture by using REST. However, both are different in that their research does not aim to increase the UI flexibility. Novak et al. [13] propose an automated testing in programming courses. Their proposal and ours are similar in that both use static code analyzers to check users' source codes. However, both are different in that their main objective is to reduce the labor of teachers in programming classes, while our main objective is to improve the quality of programming education in universities. Yulianto et al. [14] propose an automatic grader for programming assignment by using static code analyzers. Their proposal and ours are similar in that both use static code analyzers. However, both are different in that their main objective is to automate grading of submitted source codes, while our main objective is to improve the quality of undergraduates' source codes.

III. Our Proposed System

A. Motivation

In our programming education experiences in our university, there seem several understanding levels among undergraduates. The following is a part of the levels. (1) Some undergraduates think that if a source code is successfully compiled, then the source code is correct. (2) Some undergraduates think that if an executable file compiled from a source code (e.g., `a.out`) correctly runs with a few test cases, the source code is correct. (3) Some undergraduates think that if an executable file compiled from a source code (e.g., `a.out`) correctly runs with a large number of test cases, the source code is correct.

In any level, almost all undergraduates seem to share the goal, i.e., they try to write and revise a program until they are sure that their program is “correct.”

On the other hand, unfortunately, few undergraduates pay attention to the quality of their source codes such as readability and maintainability. Although the quality of a source code may not affect the execution result of the source code, ignoring the quality definitely increases the risk of having potential problems in the source code.

Through the above-mentioned experiences, we have decided to develop a system that teaches undergraduates many aspects of programming in terms of the quality in addition to the correctness.

B. Objectives

The objectives of our proposed system are to be UI-flexible and loosely-coupled as discussed later in this section. Fig. 1 shows a main difference between an ordinary design and our design. As the figure shows, the main difference is how strong the UI and the logic are coupled each other. An ordinary design is to implement a system as a part of an LMS by using APIs of the underlying LMS. However, by this design, the UI and the logic are tightly-coupled each other, thereby making it difficult to maintain themselves when the UI or the logic is required to be modified or when the underlying LMS becomes updated or obsolete. On the other hand, our design aims to be loosely-coupled by splitting the system into two layers, a flexible UI layer and a logic layer. The UI layer enables easy development of multiple UIs, and the logic layer enables easy modification of functionalities of the system. By this design, the UI and the logic can separately be modified or updated.

1) Flexible UI

To easily prepare multiple UIs for undergraduates in proportion to their understanding levels, we have decided to design and implement our proposed system to have flexible UI functionalities. This is important because, although we believe that UIs play one of the most important roles in programming education, it is difficult to provide exact one, appropriate, fixed UI for undergraduates owing to the fact that each undergraduate's programming understanding tends to vary considerably as discussed in Section I. This means that "appropriateness" of such UIs depends on each undergraduate's understanding level of programming. For example, a UI that has a large number of functionalities sometimes confuses beginners, while a too-simplified UI tends to be unsatisfactory for intermediates and seniors.

2) Loosely-coupled Architechture

To increase the maintainability of our proposed system, we have decided to design and implement the system to be loosely-coupled. This also means that the system can be used with LMSs, if needed, without heavily depending on them. This is important because heavily depending on LMSs causes the maintenance issues discussed in Section II-A.

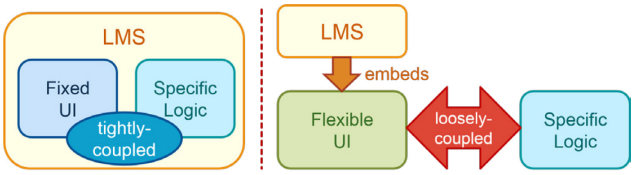

Figure 1. Difference between an ordinary design and our design. (The left is an ordinary one, and the right is ours.)

C. Overview

Fig. 2 shows a screenshot of our proposed system. As the figure shows, undergraduates learn programming by the following instructions. (Here, we assume that undergraduates already have their own programming exercises, such as "write a quick sort program in Java").
(1) Undergraduates launch their Web browsers and input the URL of the system to access it.
(2) Undergraduates input their source codes into the online source code editor on the screen.
(3) Undergraduates compile their source codes by choosing one of the compilers from the compiler list, and compile warnings and/or errors, if any, are visually reported on the screen.
(4) Undergraduates analyze their source codes by choosing one of the static code analyzers from the analyzer list, and warnings and/or errors, if any, are visually reported on the screen.
(5) Undergraduates execute their source codes with data for the standard input, if necessary, after successfully compiling them.
(6) Undergraduates follow these instructions until no warnings and errors are reported on the screen.
All the reported warnings and errors are shown as highlighted lines (the red lines in the editor) and as annotations (the marks on the left of the line numbers) on the screen.

With this system, undergraduates receive the following benefits.
(1) Undergraduates can be visually noticed that there are some potential problems in their source codes on the screen.
(2) Undergraduates can improve the quality of their source codes by revising their source codes in accordance with the reported diagnostic messages on the screen.

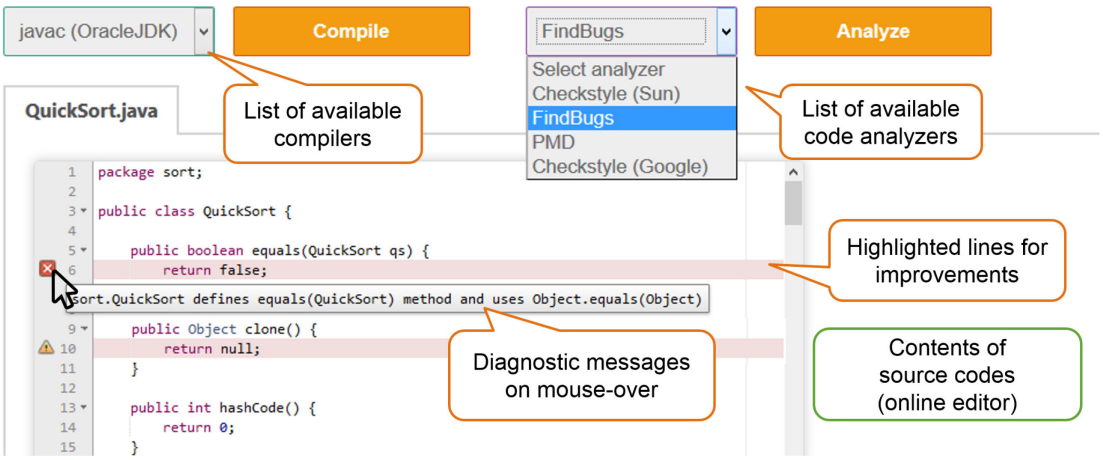

Figure 2. Screenshot of our proposed system.

D. System Requirements

Users of our proposed system need only an HTML5 compliant Web browser such as Chrome, Firefox, and Opera. They can edit, compile, analyze, and execute their source codes without installing any other software on their Web browsers.

IV. Design of Our Proposed System

A. Overview

Fig. 3 shows an overview of the design of our proposed system. As the figure shows, the system is designed as follows. (1) If an LMS is required to cooperate with the system, the LMS is required to only embed the flexible UI layer. This design reduces the number of lines of code needed for LMS-dependent plugin/extension implementation. (2) The flexible UI layer uses functionalities provided by the logic layer. Owing to the flexibility of the UI layer, we can easily modify the UI, if needed. For example, in addition to typical UI components for online programming, such as a text editor, a message window that shows messages from compilers, we can easily add a dialog box with a few lines of code. (3) The logic layer provides the functionalities needed by the UI layer, e.g., text editing, revision control, compilation, code analyzation, execution, unit testing, and so on.

B. SPA (Single-Page Application)

To cooperate with many kinds of LMSs, we have designed the whole system so that the system is fully SPA conformant. (An SPA is a Web application that properly runs without changing its URL.) This is important because LMSs generally need to control their URLs by themselves to properly run. In other words, if a plugin/extension of an LMS changes the underlying LMS's URL, the LMS could no longer be able to run properly.

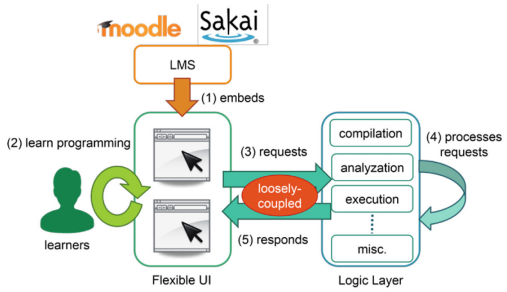

Figure 3. Overview of design of our proposed system. (The UI layer and the logic layer are loosely-coupled.)

C. REST (Representational State Transfer)

To keep our proposed system loosely-coupled, we have designed the system by using REST. Table I shows a part of the REST endpoints of the system. As the table shows, each functionality is implemented as a single REST endpoint, and JSON (JavaScript Object Notation) is used as the data format between the UI layer and the logic layer. Since the logic layer provides its functionalities by using REST, it is easy to extend its functionalities. Concretely, if a new functionality is needed, the developer is required to only add the functionality by adding a new REST endpoint in the logic layer.

V. Implementation of Our Proposed System

A. Software

We have implemented our proposed system by using the software shown in Table II. The UI layer has been implemented as an HTML5 Web application, in combination with Ace [15] as an online editor, jQuery UI [16], and jQuery [17]. The logic layer has been implemented by using Slim [18] as a PHP framework, Paris [19] as an Active Record implementation, and Idiorm [20] as an O/R mapper.

TABLE I. List of REST Endpoints of Our Proposed System

No.	Method	URI	Request MIME	Response MIME	Description
1	POST	/sources/{sourceName}	text/plain	application/json	Creates a new source code.
2	GET	/sources/{sourceId}	(empty)	application/json	Obtains a source code specified by the sourceId.
3	PUT	/sources/{sourceId}	text/plain	(empty)	Updates a source code specified by the sourceId.
4	DELETE	/sources/{sourceId}	(empty)	(empty)	Removes a source code specified by the sourceId.
5	GET	/languages/{langName}/analyzers	(empty)	application/json	Obtains the list of static code analyzers for a specific language specified by the langName.
6	GET	/languages/{langName}/compilers	(empty)	application/json	Obtains the list of compilers for a specific langauge specified by the langName.
7	GET	/analyzers/{analyzerId}	(empty)	application/json	Obtains a static code analyzer specified by the analyzerId.
8	GET	/compilers/{compilerId}	(empty)	application/json	Obtains a compiler specified by the compilerId.
9	POST	/analyzation	application/json	application/json	Analyzes source codes specified by the JSON.
10	POST	/compilation	application/json	application/json	Compiles source codes specified by the JSON.
11	GET	/analyzation/{analyzationId}	(empty)	application/json	Obtains the result of an analyzation specified by the analyzationId.
12	GET	/compilation/{compilationId}	(empty)	application/json	Obtains the result of a compilation specified by the compilationId.

TABLE II. USED SOFTWARE FOR SYSTEM IMPLEMENTATION

Software	Version	Description
Ace	1.2.2	JavaScript-based online code editor
jQuery UI	1.11.4	UI library based on jQuery
jQuery	2.1.4	JavaScript library
Slim	2.6.2	PHP micro framework that supports REST
Paris	1.5.4	Active Record implementation based on Idiorm
Idiorm	1.5.1	O/R mapper implementation in PHP
PHP	5.6.15	Server-side script language
Apache HTTPD	2.4.17	Web server
Gearman	1.1.12	Job server
MariaDB	5.5.46	RDBMS
CentOS	7.1	OS

Although the design of the system does not depend on a specific programming language, it would be better to use script languages such as PHP and JavaScript. This is because script languages enable developers to develop their applications on a trial-and-error basis with less cost when compared to non-script languages such as Java.

B. Asynchronous Processing

To increase the scalability of our proposed system, we have implemented the system to have the capability for asynchronous processing by using Gearman [21]. This is important to handle a large number of the processing requests (e.g., compilation, analyzation, execution, and so on) from its users. If the system did not have the capability, the system could soon be unresponsive to a large number of the requests from its users, owing to the shortage of computer resources such as memory space and processor time.

VI. EVALUATION OF OUR PROPOSED SYSTEM

A. Evaluation of System Usability

To evaluate the usability of our proposed system, we conducted a questionnaire survey in our class named "Object Oriented Programming," in which the Java language was used to teach programming to the members of the class. There were 42 second-year undergraduates in the class, and their Java experience is approximately half a year.

We found the following results from the questions and the answers shown in Table III.
(1) Approximately 93% (39/42) of the members are interested in improving the quality of their source codes.
(2) Approximately 98% (41/42) of the members did not know the existence of static code analyzers.
(3) Approximately 83% (35/42) of the members found some discoveries related to their source codes.
(4) Approximately 79% (33/42) of the members thought that the system helps them improve the quality of their source codes.
(5) Approximately 86% (36/42) of the members thought that the system is useful to improve the quality of their source codes.
(6) Approximately 40% (17/42) of the members did not want to use static code analyzers if the members were required to install such analyzers before using them.
(7) Approximately 79% (33/42) of the members wanted to use static code analyzers if the members were not required to install such analyzers before using them.
(8) Approximately 74% (31/42) of the members thought that it is significant to use the system in the class.

From the results, we conclude that it is important to prepare a system that facilitates use of static code analyzers with less effort, and therefore, our proposed system is a great help for undergraduates who learn programming.

B. Evaluation of UI flexibility

To evaluate the flexibility of the UI layer of our proposed system, we measured the number of lines of HTML5 code to add a new UI component and the number of lines of HTML5 code to modify the place of an existing UI component to elsewhere.

As a result, in both cases, we needed only a few lines of HTML5 code. This is because each UI component in the UI layer is represented as a single `div` element as shown in Fig. 4. The developer of the UI layer is required to only add such a code fragment into the desired position or modify the place of such a code fragment to elsewhere in the UI layer.

From the result, we conclude that the UI layer of our proposed system has enough UI flexibility to provide several kinds of UIs for its users.

TABLE III. QUESTIONS AND ANSWERS ABOUT OUR PROPOSED SYSTEM

No.	Question	Yes	No
1	Are you interested in increasing the quality (e.g., readability, maintainability, and so on) of your source codes?	39	3
2	Do you know static code anlyzers such as Checkstyle, FindBugs, and PMD?	1	41
3	Is there any discovery (e.g., about language specification) by using the system?	35	7
4	Do you think that the system help you improve the quality of your source codes in practice?	33	9
5	Do you think that the system is useful to improve the quality of your source codes?	36	6
6	Do you want to use static code analyzers even if you are required to install them on your computer?	25	17
7	Do you want to use static code analyzers if you are not required to install them on your computer?	33	9
8	Do you think it is significant to use this system in the class?	31	11

```
<div id="analyzers">
  <!--
    By JavaScript in the logic layer, this div element is
    automatically turned into a list of the available
    static code analyzers in the logic layer.
  -->
</div>
```

Figure 4. Example of HTML5 code fragment of UI component. (Each UI component is represented as a single `div` element, and therefore it is easy to add another component or to change the place of a component.)

C. Evaluation of Loose-coupling

To evaluate the degree of coupling between the UI layer and the logic layer, we measured the number of lines of JavaScript code in the UI layer and the number of lines of PHP code in the logic layer in order to add a new analyzation functionality in the logic layer. In this experiment, we prepared two systems; one is our proposed system and the other is a system that has the same functionalities as ours, but it is based on an ordinary MVC (Model-View-Controller) design.

As a result, the needed number of lines of code decreased by approximately 64% in the UI layer and decreased by approximately 30% in the logic layer, as shown in Table IV. In particular, the reduction rate of the UI layer is significant because of the UI flexibility.

From the result, we conclude that the UI layer and the logic layer are enough loosely-coupled to improve ease of development of the system.

VII. Conclusion

In this paper, we have proposed a UI-flexible, loosely-coupled programming learning system for undergraduates who learn programming. The results of our evaluation show that our proposed system is a great help for such undergraduates. In addition, we have reconfirmed that REST and SPA are indispensable to realize a UI-flexible, loosely-coupled system.

TABLE IV. Number of Lines of Code to Add New Functionality

Layer	Ordinary System	Our Proposed System	Reduction Rate (%)
UI	118	42	64.4
Logic	275	194	29.5

References

[1] Moodle, https://moodle.org/.

[2] Sakai, https://www.sakaiproject.org/.

[3] A. Sureka, Y. R. Reddy, P. Muenchaisri, and M. Tsunoda, "A report on Software Engineering Education Workshop 2014 co-located with Asia-Pacific Software Engineering Conference 2014," ACM SIGSOFT Software Engineering Notes, vol. 40, issue 1, pp. 40–43, January 2015.

[4] T. Hayakawa, C. Nishikado, and T. Hikita, "A proposal for loosely-coupled programming learning system for undergraduates," Proc. of 6th Annual International Conference on Computer Science Education: Innovation & Technology (CSEIT 2015), pp. 174–177, Octber, 2015.

[5] Checkstyle, http://checkstyle.sourceforge.net/.

[6] FindBugs, http://findbugs.sourceforge.net/.

[7] PMD, http://pmd.sourceforge.net/.

[8] codepad, http://codepad.org/about.

[9] Ideone, https://ideone.com/.

[10] VirtualPairProgrammers.com, https://www.virtualpairprogrammers.com/.

[11] K. Jelemenska and P. Cicak, "SystemC modeling skills assessment in Moodle environment," Proc. of IEEE 17th International Conference on Intelligent Engineering Systems, pp. 351–356, June 2013.

[12] K. Itou, Y. Mima, and A. Ohnishi, "A linkage mechanism between course-management-system and coursework-checking-functions over web services," IPSJ Journal, vol. 52, No. 12, pp. 3121–3134, December 2011, (in Japanese).

[13] M. Novak and M. Binas, "Automated testing of case studies in programming courses," Proc. of 9th International Conference on Emerging eLearning Technologies and Applications, pp. 157–162, October 2011.

[14] S.V. Yulianto and I. Liem, "Automatic grader for programming assignment using source code analyzer," Proc. of International Conference on Data and Software Engineering, pp. 1–4, November 2014.

[15] Ace, https://ace.c9.io/.

[16] jQuery UI, https://jqueryui.com/.

[17] jQuery, https://jquery.com/.

[18] Slim, http://www.slimframework.com/.

[19] Paris, https://github.com/j4mie/paris.

[20] Idiorm, https://github.com/j4mie/idiorm.

[21] Gearman, http://gearman.org/.

AUTHOR'S PROFILE

Tomokazu Hayakawa was born in Ichikawa, Japan in 1982. He earned B.Sc., M.Eng., and D.Eng. degrees in computer science from Meiji University, Kawasaki, Japan, in 2004, 2007, and 2013, respectively. He joined TG Information Network Co., Ltd., where he worked as a computer engineer. Since 2014, he has been an assistant professor at Dept. of Computer Science of Meiji University, Japan. His research interests include Web applications (especially RIA technologies), software engineering, and e-learning. He is a member of the Information Processing Society of Japan.

Chika Nishikado was born in Ohtawara, Japan in 1993. She earned B.Sc. degree in computer science from Meiji University, Kawasaki, Japan, in 2015. Since 2015, she has joined RAKUS Co., Ltd., where she has worked as a computer engineer. Her main research interest is e-learning.

Teruo Hikita (M'79) was born in Nishinomiya, Japan in 1947. He earned B.Sci in 1970, M.Sci in 1972, and D.Sci in 1978, all from University of Tokyo. He was at University of Tokyo and Tokyo Metropolitan University, and since 1989 he is at Dept. of Computer Science of Meiji University in Kawasaki, Japan. His current research interests are in Web technologies. Professor Hikita is a member of ACM, IPSJ and SIAM.